

a quarterly bulletin
of the IEEE computer society
technical committee
on

Database Engineering

Contents

Letter from the Editor	1
Databases and Natural Language Processing <i>Z. W. Pylyshyn and R. I. Kittredge</i>	2
TEAM: An Experimental Transportable Natural Language Interface <i>P. Martin, D. E. Appelt, B. J. Grosz, and F. Pereira</i>	10
A Multilingual Interface to Databases <i>H. Lehmann, N. Ott, and M. Zoepritz</i>	23
Evaluation and Assessment of a Domain-Independent Natural Language Query System <i>M. Jarke, J. Krause, Y. Vassiliou, E. Stohr, J. Turner, and N. White</i>	34
Modelling Natural Language Data for Automatic Creation of a Database from Free-Text Input <i>N. Sager, E. C. Chi, C. Friedman, and M. S. Lyman</i>	45
Alternatives to the Use of Natural Language in Interfacing to Databases <i>Z. Pylyshyn</i>	56
Menu-Based Natural Language Interfaces to Databases <i>C. W. Thompson</i>	64
Calls for Papers	71

Special Issue on Natural Language and Databases

**Chairperson, Technical Committee
on Database Engineering**

Prof. Gio Wiederhold
Medicine and Computer Science
Stanford University
Stanford, CA 94305
(415) 497-0685
ARPANET: Wiederhold@SRI-AI

**Editor-in-Chief,
Database Engineering**

Dr. David Reiner
Computer Corporation of America
Four Cambridge Center
Cambridge, MA 02142
(617) 492-8860
ARPANET: Reiner@CCA
UUCP: decvax!cca!reiner

**Associate Editors,
Database Engineering**

Dr. Haran Boral
Microelectronics and Computer
Technology Corporation (MCC)
9430 Research Blvd.
Austin, TX 78759
(512) 834-3469

Prof. Fred Lochovsky
Department of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S1A1
(416) 978-7441

Dr. C. Mohan
IBM Research Laboratory
K55-281
5600 Cottle Road
San Jose, CA 95193
(408) 256-6251

Prof. Yannis Vassiliou
Graduate School of
Business Administration
New York University
90 Trinity Place
New York, NY
(212) 598-7536

Database Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Database Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, database architecture, database machines, intelligent front ends, mass storage for very large databases, distributed database systems and techniques, database software design and implementation, database utilities, database security and related areas.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews, summaries, case studies, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Database Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in the Database Engineering Technical Committee is open to individuals who demonstrate willingness to actively participate in the various activities of the TC. A member of the IEEE Computer Society may join the TC as a full member. A non-member of the Computer Society may join as a participating member, with approval from at least one officer of the TC. Both full members and participating members of the TC are entitled to receive the quarterly bulletin of the TC free of charge, until further notice.

Letter from the Editor

The term "natural language" has certainly generated controversy in the database area. Even taking aside the staunch supporters and opponents of natural language as an interface to databases, we have seen waves of praise, hope, and promise, followed by disappointments and condemnations.

I believe that the relationship between natural language and databases is now in calmer seas - we are seeing an upswing of interest in natural language and much research activity. This new interest may be explained by three recent developments: (1) the technical improvements of natural language systems following knowledge base technology, (2) the consideration of natural language not only in isolation as a query language but also in combination with other forms of interfaces (e.g., menus), and (3) the commercialization of natural language - always a strong indicator of research interest.

This issue of DBE is on Natural Language and Databases. It investigates not only natural language as a query language, but also free-text analysis and mapping of text into databases. A large number of research projects and development efforts using natural language in conjunction with databases are currently under way in North America and Europe. The goal of this issue is to collect and present some representative work from both continents, from both industry and academia, and for both natural language processing and natural language system evaluation.

The first article, *Databases and Natural Language Processing* by Zenon Pylyshyn and Richard Kittredge, introduces the topic and points to the major research projects. This article is followed by descriptions of two systems which are in advanced development stages. First, Paul Martin et al describe the project TEAM at SRI International (*TEAM: An Experimental Transportable Natural Language Interface*), a state-of-the-art natural language query system. Second, Hubert Lehmann et al present the USL project at IBM Heidelberg (*A Multilingual Interface to Databases*), a research effort that uses a more global definition of natural language (not only English!). The latter system has been the subject of extensive empirical evaluations, the results of which are summarized in the article by Matthias Jarke et al (*Evaluation and Assessment of a Domain-Independent Natural Language Query System*). Mapping English text in technical domains (e.g., medicine) into a database for further processing is the topic of the article by Naomi Sager et al (*Modeling Natural Language Data for Automatic Creation of a Database from Free-Text Input*). To put things into perspective, limitations of current natural language systems, as well as two suggestions for future research directions to overcome some of these limitations, are given in *Alternatives to the Use of Natural Language in Interfacing to Databases*, by Zenon Pylyshyn. One of these research directions is exemplified by the last article of the issue (*Menu-Based Natural Language Interfaces to Databases*) by Craig Thompson.

I wish to thank all the authors of this DBE issue for accepting my invitation, for the time they devoted to produce quality contributions, and for meeting all deadlines with no complaints.

Yannis Vassiliou

July 1985.

Databases and Natural Language Processing

Zenon W. Pylyshyn, University of Western Ontario, London, Canada

Richard I. Kittredge, Universite de Montreal, Montreal, Canada

Progress in the computer analysis of natural language (NL) text offers a number of promising new directions in database design. For example, the use of unrestricted NL queries to interrogate databases offers an attractive option to artificial query languages or menus -- especially for nontechnical users. Recent successes in developing such "front-ends" to databases represent an important commercial application of NL processing. Other potential applications are also briefly examined, including automatic text analysis for indexing, abstracting and formatting of textual information. Several accomplishments and shortcomings of this technology are sketched.

1. General Introduction

Databases for general office, management and consumer use, present special problems both in terms of challenging computer science techniques for dealing efficiently with large databases and in terms of the design of user interfaces. Because such databases are intended to be used by nontechnical people it is crucial that accessing these databases be convenient and natural, or at least easy to learn. One of the largest obstacles to the widespread acceptance of consumer and management databases is the resistance of the average user to the relatively cumbersome method of access, or at least to the perceived rigidity of the interface between the user and the stored information. In this overview we will consider some actual and potential contributions of Artificial Intelligence technologies to the alleviation of some of these difficulties, with particular regard to developments in natural language processing.

A slogan in the commercial use of artificial intelligence is that we must make the machine know more about the user so that the user will need to know less about the machine. This slogan highlights an important general point, namely that if a user is to continue to operate the way he or she normally would, then the machine will have to adapt to that way. Since the usual way that we seek information is by asking questions in our native language, this implies that a natural language query system may be the most natural way to access information. Furthermore, since a great deal of the information that we need is in the form of natural language text, the analysis of such text could be an important component of database processing. Below we examine a number of developments in the processing of natural language, with a view to its relevance to database technology.

2. Natural Language as a Database Query Interface

[WOOD83] presents some persuasive arguments for the importance of natural language as a communication channel between man and machine. They are based on the observation that (1) People already know natural language, so they do not need to bear the burden of learning an artificial language nor of remembering its conventions over periods of disuse, and (2) Using a natural language spares the user from having to

translate his requests from the form in which they presumably occur to him into a restricted artificial form. These two reasons alone can be the bases of a major justification for developing natural language interfaces. Even when users have the time and patience to learn an artificial language, and even when they become experts in the use of an artificial language, these two reasons remain important. Even with experienced users there arise occasions when they know what they want the machine to do but cannot recall how to express it in the artificial language, or find it difficult to do so, or attempt it and make errors. Furthermore, even in those cases where the user does remember how to express the query in an artificial language, and can do so with little error, the mismatch between the conceptual structure of a computer query system and a human natural conceptualization of problems and intentions presents a serious problem which leads users to prefer to consult with a human interlocutor -- even when that course appears inefficient -- than deal with the conceptualization of the machine. This is especially true when the data being interrogated are intrinsically natural language data.

Woods argues that the fundamental difficulty with artificial query languages does not lie in their superficial syntactic form, but in their underlying conceptual structure -- e.g. their failure to use devices such as anaphora, ellipses, metalinguistic references -- in other words, just the sorts of constructions that typically make natural language processing difficult. Many (e.g. [HAYE81], [COHE81] have also made similar points. As a consequence, some have suggested that artificial languages or a restricted subset of natural languages should preserve the important conceptual properties of natural language (e.g. [HAYE81]).

The use of natural language to query databases is not without its problem, however, especially if the language analysis system is limited. Some difficulties with the use of natural language and several alternative interface strategies are discussed in the articles in this issue by Pylyshyn and by Thompson.

2.1. State of the Art

The use of natural language to interrogate databases has been one of the most successful and most visible areas of application of artificial intelligence in recent years. The commercial success of products such as INTELLECT, which is currently being marketed by IBM (see [ARTI81]; [HARR77]), ENGLISH and Francais (Natural Language front ends to the RAMIS II database, Marketed by Mathematica Products Group), Themus (a Natural Language front end to the Oracle database system which has a learning capability -- marketed by MBS) and products being developed for personal computers by companies like Symantec, has made many people look to such interface systems as a potential answer to the problem of allowing computer-naive consumers access to large-scale databases.

Current natural language systems not only have the capability of answering complete self-contained grammatical questions, but in some cases can also understand user inputs containing simple pronoun references to words in earlier queries, inputs with misspelled words or minor grammatical errors, certain cases of ellipses (queries that are incomplete and rely on reuse of words from a previous query -- e.g. *How many grocery stores are there? Hardware stores?*), and certain definitions introduced by the user. Current systems allow only limited updates of the database by the user in interaction

with the Natural Language system, incorporate only a very limited theory of the domain of application, do not translate the query into a general logical form from which inferences can be carried out, and in general are not capable of analysis at the level of discourse pragmatics, which requires that the system maintain a model of the user's needs and intentions. [HEND82] calls such systems 'level 1' systems.

While current 'level 1' systems are broader in the range of queries they can accept than the research systems of 10 years ago (e.g. [WOOD72], [WINO72]), most of them are, in fact, based on grammatical and parsing ideas that differ little from those early systems. Indeed, most of them use parsers based on the augmented recursive transition network system developed by Woods, Kaplan and others (see [WOOD72]). They accomplish their more impressive performance by narrowing their domain of application. As well as using a separate grammatical module (a highly desirably architectural feature which makes it easier to change and fine-tune the system to different applications), they generally make heavy use of the lexicon in order to add a variety of tricks that apply in limited domains. Such devices can be used, for example, in order to resolve certain types of anaphoric reference as well as to eliminate certain potential ambiguities. In addition, most of these systems require some customization for specific databases. This is the case, for example, in the INTELLECT, which requires a customized module for mapping entries in its lexicon directly onto data fields.

Even the best current commercial systems are poor at handling expressions with two or more quantifiers (*Does every shop supervisor earn more than any of the craftsmen who works under him?*). In addition, they do not contain a model of the user. Some such model is necessary to deal sensibly with a variety of queries -- for example, in order to correctly handle questions which result in a null answer (e.g. if asked *Do union members earn more than non-union workers?* when all workers in a certain company are either unionized or none of them are, a system which had no representation of what a user needed to know would simply provide the unilluminating answer *no*).

Several substantial level 1 systems are in the advanced prototype state. Among the better-known ones are the following:

- The TQA system, under development at Yorktown Heights since the early 1970's, has undergone a constant evolution, but is still based on a transformational parser developed by Petrick and Plath. During 1978-79 the system was given an extensive test by the White Plains municipal office for querying their database on zoning and land use. Statistics collected during that trial [DAME81] showed that some 65% of the 800 queries to the system were correctly parsed and answered. Users sometimes had to reformulate a query to stay inside the artificial limits of the system's syntax and vocabulary (a typical problem for present query systems).
- The USL system at IBM-Heidelberg represents about the same degree of advancement as the TQA system, although it uses a different parser and semantic approach. Its market advantage lies in the fact that there exists a version for German as well as for English, Italian, French and Spanish (see the article in this issue).
- The ASK system is being developed at the California Institute of Technology [THOM83] for commercialization by Hewlett-Packard Corporation. ASK uses semantic networks to give a simple knowledge representation of the database domain. In addition to rapid parsing and analysis, its features include a facility for tailoring an existing database to a particular user's 'Context' through an interactive dialogue. This includes the ability to add new definitions and extend the database structure through dialogues.

The only large scale working systems are level 1. Many research systems contain significant improvements over commercial level 1 systems, and there are also fragments of level 2 designs in various stages of development. These will be mentioned briefly in section 4. Below we discuss some applications of developments in natural language processing for other than providing a natural language query capability.

3. Natural Language for Updating and Maintaining a Database

A major problem arises in natural language 'updates' to databases. Even though natural language is not necessarily the most convenient medium for bulk data entry, it is important to have some facility for making limited changes. At the very least, one wants to be able to add or modify individual facts. But unless very carefully controlled, natural language updates are potentially dangerous. The potential ambiguity of update commands may not be obvious to the user, and allow damage to data which is hard to undo.

In addition to such on-line updating capabilities, a major area of research involves the preparation of natural language text for inclusion in a database. This requires the analysis of extended text to extract its meaning so that efficient database techniques and indexing methods can be applied. Systems which analyze extended text usually cannot be interactive, since the author of the text may not be on-line. In any case, the demands of high volume processing normally make interaction prohibitive. Because of this, extended text systems must usually be richer in linguistic detail, since there is no 'second chance' to rephrase the input.

One of the most significant advances in text analysis over the past decade has been the refinement of techniques for mapping texts from specialized subject areas into 'information formats', which are tabular representations of the data contained in the texts. These 'informatting' techniques have grown out of work done at New York University (e.g., [SAGE78]) which has concentrated on scientific and technical writing in medicine and related fields. This work has several applications for information science. One of the most important ones is in creating a database from full text.

For example, [HIRS82] report on the conversion of hospital discharge summaries, written by an attending physician in telegraphic style, into a relational database. This access to information contained in the text opens up a new source of medical data for statistical analysis. [GRIS78] also reports on the use of such techniques for query systems, where the query can be processed into semantic form using the same techniques (more details of this work are given in the article by Chi et. al. in this issue). Central to this approach is a detailed linguistic study of the particular technical 'sublanguage'.

Although a number of experiments have been carried out on converting sublanguage texts to information formats, this technique appears to be at least a few years from substantial commercial application, at least for complex medical texts. The reason for this is that while a large percentage of sentences in a typical report can be mapped into a structured format, not all sentences can be formatted. In part, this is due to the fact that even technical reports will typically contain material which lies outside the particular sublanguage for which the system was specialized (e.g., remarks on the personal history of the patient and his family in a hospital record). Because of

this one needs a much larger grammar and lexicon, perhaps one that begins to approach that of the language as a whole.

One of the more ambitious goals in the area of text analysis, and one that could potentially have a large impact on database design, is automatic abstracting. Much of the work on this problem was carried out a number of years ago, and hence does not use state-of-the-art techniques. However, there are several recent revivals of interest, which approach the problem from quite different perspectives. One is some recent work at the U.S. Naval Research Laboratories on the automatic dissemination and summarization of telegraphic messages concerning malfunctioning electronic equipment on board ships at sea. A system has constructed a system which uses the NYU string parser and sublanguage techniques to convert paragraph-length messages into information formats. Format entries are analyzed for revealing combinations of semantic classes, leading to the choice of one entry (the equivalent of a single proposition) which best summarizes the whole paragraph. The NRL team has built a prototype system which successfully produces single-sentence summaries for many of the simpler paragraphs, though its performance is at present very limited. It appears that much more research is needed on the linguistic problems of telegraphic sublanguages.

Another approach to abstracting, is the work on summarizing news reports, carried out by R. Schank and a number of his former students from Yale (e.g., [DEJO79]). They have used 'sketchy scripts' to represent the structure of stereotypical events and their subevents. The hierarchical structure of scripts allows a summarization (on the topmost level) of a story which has been 'understood' (i.e., matched) according to the script representation. This approach has only been applied in very limited domains at present and its generalizability to less restricted text is open to debate. One interesting recent application of these ideas is the NOMAD system at the University of California at Irvine [GRAN83]. NOMAD is designed to analyze telegraphic ship-to-shore messages in 'command and control' situations. The system uses script-based expectations to interpret messages and paraphrase them into full standard English. Specific 'syntactic' patterns of the sublanguage are also used. This system is still in the early experimental stage.

4. Research Issues in Natural Language Analysis

Level 1 systems can sometimes be improved in a number of ways without requiring representation of very large amounts of general knowledge of the domain and the user -- as would be required for higher level systems. For example, one of the most promising techniques for allowing natural language interfaces to be transported to new database domains (with their associated differences in input vocabulary) is to have the system acquire this linguistic information during a dialogue with a database administrator who has no knowledge of computational linguistics. The TEAM system at SRI [GROS83] (see also the description in this issue) has an acquisition component which queries the database administrator about the data types to automatically set up a grammar and dictionary usable by the interface component. Another improvement, still in the research stage, is a facility for providing 'concise responses', so that instead of answering a question like "Who drives a company car?" with a list of people (an extensional reply), the system would give a more meaningful response (the intensional reply) such as: "The president and the vice-presidents".

Current operational systems do not employ either an explicit, detailed representation of the knowledge associated with the application domain, or a model of the user's goals, state of knowledge, and limitations. [HEND82] have called systems with extensive explicit domain knowledge 'level 2' systems and systems with a detailed model of the user (in addition) 'level 3' systems. A good deal of direct research is taking place on modelling such systems or on the underlying problems of representing the linguistic and extralinguistic knowledge which they require.

A number of experimental systems which incorporate level 2 capabilities are now under construction. Representative of these are the IRUS system from BBN

the KNOBS system [PAZZ83] under development at MITRE Corporation, and the HAM-ANS system from Hamburg. KNOBS makes use of several knowledge sources during the processing of a query, including scripts with stereotypical knowledge of the particular domain and inferencing rules for explicating information which is missing from the user's input. Within the context of the problem domain (an expert system providing consultant services to an Air Force tactical air mission planner), KNOBS illustrates the feasibility of integrating several different kinds of knowledge-based processing in a natural language interface. The HAM-ANS system, being developed at the University of Hamburg, also uses several different knowledge sources. It is an attempt to design a "core" natural language interface to three different background systems: an expert system, a vision system, and a database system [HOEP83].

Some preliminary attempts are being made to integrate a (partial) model of the user into natural language interfaces to query systems. A project at the University of California at Berkeley is aimed at building a consultant ('UC') for the UNIX operating system. In particular, UC provides an analysis of the user's goals during interaction with the system, employing rules ('frames') of considerable generality. For an overview of UC, see [WILE82].

A good deal of research is being conducted at several major American centers on knowledge representation and discourse pragmatics, with the specific intention of extending the performance of natural language interfaces. For example, the University of Pennsylvania is carrying out a study of Flexible Communication with Knowledge Bases, with a strong emphasis on discourse pragmatics. One of the features of this research will be to acquire an integrated view of both linguistic and visual communication with databases. This requires a representation of certain types of knowledge which will interface with both linguistic structures and with two and three-dimensional images. This research has also emphasized the recognition of various kinds of user misconceptions on the basis of rules for goal-oriented linguistic behavior.

Despite the acknowledged commercial successes of level 1 systems, and the encouraging research on level 2 systems, there are reasons for thinking that in the short and perhaps even medium term (5-10 years), Natural Language systems may not be the best solution for making consumer databases widely available and convivial. Problems of interpreting queries have only been solved in an ad hoc way for very narrow relational databases, and the customization of such natural language query systems to new subject areas (new databases) represents a serious investment of time and effort, assuming it is possible at all. A large number of problems have to be solved before such systems can be considered useful for the general consumer, many of which have to do with low-level problems associated with the use of the keyboard. The tedium of typing

suggests the importance of allowing abbreviations (and even automatic word-completions), providing rapid on-line spelling correction, dictionary maintenance (including facilities for defining new macro-expansions based on function keys and special keyboard aids) as well as helpful on-line syntax checking, ambiguity reduction and other help facilities. The resistance to the use of keyboards also emphasizes the importance of exploring other possible modes of input, including speech and pointing devices.

In addition, as we have already suggested, development of the sort of natural language system that would be truly useful raises a host of deep problems that are currently under investigation -- such as that of assigning anaphoric reference to general terms and pronouns, interpreting fragmentary and ungrammatical queries, recovering the presuppositions of questions, determining the meaning and scope of quantifiers (such as "some", "most", "none", "all") and negation, and interpreting indirect "speech acts" (such as "I need to know...") or metalinguistic assertions (such as "No, I meant the most recent figures," as a response to the data reported when the system was asked for trends in the price of certain commodities.)

4.1. Location of Natural Language research

Most of the long-term frontier research in natural language processing is being carried out in large research laboratories specializing in Artificial Intelligence. These include laboratories universities such as Pennsylvania, Stanford, Carnegie-Mellon, MIT, New York or Yale in the USA; Marseille, Hamburg, or Edinburgh in Europe; or Toronto, Simon Frazer, Montreal or Western Ontario in Canada. The smaller institutions typically specialize in particular problems associated with natural language processing (for example, the Canadian universities tend to focus on problems of knowledge representation). Among nonacademic institutions, significant research in natural language processing is being carried out at SRI International, Bolt Beranek and Newman, Bell Laboratories, Xerox, IBM and Hewlett-Packard. One of the largest and most ambitious basic research projects is being pursued at the Center for the Study of Information and Language, a consortium of research laboratories centered at Stanford. A considerable amount of work has also been done on the natural language problems implicit in machine translation (e.g. the TAUM project at the Universite de Montreal, the Eurotra project being carried out by the European Economic Community, or the machine translation projects in Japan).

REFERENCES

- [ARTI81] Artificial Intelligence Corporation. *INTELLECT User's Manual*. Waltham, Mass., 1981.
- [COHE81] Cohen, P., Perrault, C., and Allen, J. "Beyond question-answering", Technical Report No. 4644, Bolt Beranek and Newman Inc., May, Cambridge, Mass., 1981.
- [DAME81] Damereau, F. "Operating Statistics for the Transformational Question Answering System." *American Journal of Computational Linguistics*, 7:1, 30-42, 1981.
- [DEJO79] Dejong, G. *Skimming Stories in Real Time: An Experiment in Integrated*

- Understanding*. Res. Rep. No. 158, Yale Computer Science Department, 1979.
- [GRIS78] Grishman, R., and Hirschman, L. "Question Answering from Natural Language Data Bases". *Artificial Intelligence*, 11:25-43, 1978.
- [GRAN83] Granger, R., Staros, C., Taylor, C., and Yoshii, R. "Scruffy Text Understanding: Design and Implementation of the NOMAD System". *Pros. of the Conf. on Applied Natural Language Processing*, Santa Monica, 1983.
- [GROS83] Grosz, B. TEAM: Transportable Natural Language Interface System. *Pros. of the Conf. on Applied Natural Language Processing*, Santa Monica, 1983.
- [HARR77] Harris, L. User oriented data base query with the ROBOT natural language query system. *Int. J. Man-Mach. Stud.*, 9:6 (November), 697-713, 1977.
- [HAYE81] Hayes, P.J. "Anaphora for Limited Domain Systems", *Proc. Seventh International Joint Conference on Artificial Intelligence*, Vancouver, 416-422, 1981.
- [HEND82] Hendrix, G., et. al. "Natural Language Interface." *American Journal of Computational Linguistics*, 8:2, 56-61, 1982.
- [HIRS82] Hirschman, K., and Sager, N. Automatic Informatting of a Medical Sublanguage. In Kittredge, R. and Lehrberger, J. (eds.) *Sublanguage: Studies of Language in Restricted Semantic Domains*, de Gruyter, 1982.
- [HOEP83] Hoepfner, W. et. al. "Beyond Domain Independence: Experience with the development of a German language access system to highly diverse background systems". *IJCAI-83*, Karlsruhe, 1983.
- [PAZZ83] Pazzani, M., and Engelman, C. Knowledge-Based Question Answering. *Proc. of the Conf. on Applied Natural Language Processing*, Santa Monica, 1983.
- [PYLY85] Pylyshyn, Z. "Alternatives to the Use of Natural Language in Interfacing to Databases", *Database Engineering*, this issue, 1985.
- [SAGE78] Sager, N. "Natural Language Information Formatting: The Automatic Conversion of Texts to a Structured Data Base". In M.C. Yovits, (Ed.), *Advances in Computers*, 17, 89-162, New York: Academic Press, 1978.
- [THOM83] Thompson, B., and Thompson, F. Introducing ASK, a Simple Knowledgeable System. *Proc. of the Conf. on Applied Natural Language Processing*, Santa Monica, 1983.
- [WILE82] Wilensky, R. Talking to UNIX in English: an Overview of UC. *Proc. of the 2nd AAAI Conf.*, 1982.
- [WINO72] Winograd. *Understanding Natural Language*. New York: Academic Press, 1972.
- [WOOD83] Woods, W. "Natural Language Communication with Machines: An Ongoing Goal:", *Technical Report No. 5375*, Bolt Beranek and Newman, Cambridge, Mass., July, 1983.
- [WOOD72] Woods, W., Kaplan, R., and Nash-Webber, B. *The Lunar Sciences Natural Language Information System: Final Report*, Bolt, Beranek and Newman, TR 2378, Cambridge, Mass., 1972.

TEAM: An Experimental Transportable Natural-Language Interface

*By Paul Martin, Douglas E. Appelt, Barbara J. Grosz, Fernando Pereira
Artificial Intelligence Center
SRI International*

ABSTRACT

This paper is a brief description of TEAM, a project whose goal was to design an experimental natural-language interface that could be transported to existing database systems by people who already possessed expertise in their use. In presenting this overview, we have concentrated on those design aspects that were most constrained by the requirements of transportability.

1 A Functional Description

A natural-language interface (NLI) to a computer database provides users with the capability of obtaining information stored in the database by querying the system in a natural language (e.g., English). The use of natural languages as a means of communication with computer systems allows users to frame a question or a statement in the way they think about the information being discussed, thereby freeing them from the need to know how the computer stores or processes the information. However, most existing NLI systems have been designed specifically to treat queries that are constrained in three ways: (1) they concern a single application domain; (2) they pertain to information in a single database; (3) they handle only a single task, namely, database query.¹ Constructing a system for a new domain or database requires a new effort almost equal to the original one in magnitude.

Transportable NLIs that can easily be adapted to new domains or databases are potentially much more useful than domain- or database-specific systems. However, because many of the techniques already developed for custom-built systems preclude automatic adaptation of the systems to new domains, the construction of transportable systems poses a number of technical and theoretical problems. In describing the transportable NLI system called TEAM (Transportable English database Access Medium), that was the focus and objective of a four-year project, this article emphasizes those choices in system design imposed by the requirement of transportability.² For some problems, the design decisions incorporated in TEAM are generally applicable to a wider range of natural-language processing systems; for others, we were forced to take a more limited approach.

1.1 Transportability

One of the major challenges faced in building NLIs is to provide the information needed by the system to bridge the gap between the way the user thinks about the domain of discourse and the way the computer handles the information it possesses about the domain. Existing databases employ

¹This constraint is more limiting in many ways than the other two. For example, queries are typically treated largely in isolation; very few features of dialogue are handled. Since this remains a constraint in TEAM it will not be discussed further in this article.

²Space limitations have compelled us to omit many of the specific problems faced in this research; for a fuller treatment, please see the journal article [Gros85].

different representational conventions, many of which favor storage efficiency over perspicuity. For example, one might encode geographic information about mountain peaks in Switzerland as part of a file of information about the mountain peaks of the world, identifying them with a "SWZ" in a COUNTRY field, or using a SWISS? feature field for which a "Y" indicates that a peak is in Switzerland and an "N" indicates it is not. Or the information might reside in a separate file on Switzerland, or one on Swiss mountain peaks. The kinds of queries a user might pose—for example "What is the highest Swiss peak?" "Are there any peaks in Switzerland higher than Mt. Whitney?" "Where is the Jungfrau?"—are equally appropriate for all the aforementioned encodings and the inputs to the NLI (an English query) remain unchanged. The output (commands to a database system), however, will be quite different. One of the main functions of the NLI is to make the necessary transformations, thus insulating the user from the particularities of the database structure.

To provide this insulation and to bridge the gap between the user's view and the system's structures requires a combination of domain-specific and general information. In particular, the system must have a model of the subject matter of the application domain. Included in this model will be information about the objects in the domain, their properties and relationships, and the words and phrases used to refer to each. Finally, the system must know the connection between entities in that model and the information in the database. A major challenge in constructing transportable systems is to provide a means for easy acquisition of domain-specific information.

TEAM is one of several recent attempts to build transportable systems (some of which are described elsewhere in this issue.) Different approaches to transportable systems reflect diverse conceptions of the kinds of skills and knowledge that might be required of those who will be doing the adaptations (in particular, whether they must have expertise in natural-language processing), and what parts of the system might change (in particular, whether the database can be restructured to fit the requirements of the NLI).

A major hypothesis underlying TEAM may be stated as follows: if an NLI is constructed in a sufficiently well-principled manner, the information needed to adapt it to a new database (and its corresponding domain) can be acquired from users who have general expertise about computer systems and the given database, but who do not have any special knowledge about natural-language processing or this NLI.

In testing this hypothesis, we also assumed (for both theoretical and practical reasons) that the database could not be restructured. Theoretically, it is the most conservative choice we could have made; it imposed general solutions upon certain issues of system design, because we could not restructure the data to alleviate problems of natural-language processing. Such restructuring can often bring about a closer match between the way information is stored and the way it is referred to in NL expressions. For instance, in the previous example, a database structure that includes the SWISS? feature field is more difficult to handle in a general manner than one that uses the COUNTRY field encoding. From a practical standpoint, the choice reflected our desire to provide techniques adequate to handle existing databases, some of which are quite large and complex, hence fairly difficult to restructure.

1.2 Using TEAM

The TEAM system is designed to interact with two kinds of users: a *database expert* (DBE) and an *end user*. The DBE engages in an acquisition dialogue with TEAM to provide the information needed to adapt the system to a new database, and, when desired, to expand its capabilities in answering questions about a database (e.g., by adding new verbs or synonyms for existing words). Once a DBE has provided TEAM with the information it needs about a database and domain, any

WORLD				
NAME	CONTINENT	CAPITAL	AREA	POP
Afghanistan	Asia	Kabul	260,000	17,450,000
Albania	Europe	Tirana	11,100	2,620,000
Algeria	Africa	Algiers	919,961	18,510,000

PEAK			
NAME	COUNTRY	HEIGHT	VOL
Aconcagua	Argentina	23,080	N
Annapurna	Nepal	26,504	N
Chimborazo	Ecuador	20,702	Y

CONT			
NAME	HEMI	AREA	POPULATION
Africa	S	11,600,000	41,200,000
Antarctica	S	6,000,000	600
Asia	N	16,990,000	2,366,000,000

BCITY		
NAME	COUNTRY	POP
Brussels	Belgium	1,050,787
Buenos Aires	Argentina	8,925,000
Canberra	Australia	210,600

Figure 1: Sample Database

number of end users can use the system to query the database.

The TEAM system thus has two major modes: acquisition and question-answering. The acquisition dialogue with the DBE is oriented around the database structure. It is a menu-driven interaction through which the DBE provides information about the files and fields in the database,³ the conceptual content they encode and how they encode it, and the words and phrases used to refer to these concepts. Hence the DBE must know about the particular database structure and the subject domain its information covers, but he does not need to know how TEAM works or any special language-processing terminology.

The question-answering system consists of two major components: (1) the DIALOGIC system [Gros82] for mapping natural-language expressions onto formal logical representations of their meanings; (2) a schema translator that transforms these representations into statements of a database query language. DIALOGIC and the schema translator require both domain-specific and domain-independent information. The requisite domain-independent information is part of the core TEAM system; the domain-specific information is obtained by the acquisition component.

1.3 A Sample Database

We will use the database shown schematically in Figure 1 to help illustrate various aspects of TEAM. This database comprises four files (or, relations) of geographic data. The first file, WORLD, has five fields—NAME, CONTINENT, CAPITAL, AREA and POP; respectively, they specify the continent, capital, area, and population for each country in the world. Various mountains in the world are represented in the second file, named PEAK, along with their country, height, and an indication as to whether they are volcanic. The third file, named CONT, shows the hemisphere, area, and population of the continents. The fourth file, BCITY, contains the country and population of some of the larger cities of the world. Because several files may have fields with the same names, TEAM prefixes file names to field names to form unique identifiers (e.g., WORLD-NAME, PEAK-NAME, CONT-POP, BCITY-POP); we will do likewise in our discussion.

TEAM distinguishes among three different kinds of fields: feature, arithmetic, and symbolic. *Feature fields* contain true/false values indicating whether or not some attribute is a property of the file subject. PEAK-VOL and CONT-HEMI are feature fields. *Arithmetic fields* contain numeric values on which computations (e.g., averaging) can be performed WORLD-AREA and PEAK-HEIGHT are examples of arithmetic fields. Let us note, however, that a field containing social security numbers

³TEAM currently assumes a relational database with a number of files. No difficult language-processing problems would result from conversion to other models.

would be treated more naturally as a symbolic field than as an arithmetic field, because it is unlikely that any arithmetic computations would be done on such numbers. *Symbolic fields* typically contain values that correspond to nouns or adjectives denoting the subtypes of the domain denoted by the field. `WORLDC-NAME` and `PEAK-COUNTRY` are examples.

More information can be gleaned from a database than simply what the individual files contain. For instance, the continent on which a peak is located can be derived from the country in which it is located and the continent of the country. Likewise, the hemisphere in which a country is located can be determined from the continent on which the country is located and the hemisphere of that continent. `TEAM` allows the `DBE` to specify *virtual relations* that convey such additional information.

2 The `TEAM` System Architecture

The design of `TEAM` reflects several constraints imposed by the demand for transportability; our discussion will emphasize those aspects of the design. The need to decouple the representation of what a user means by a query from the procedure for obtaining that information from the database obviously affected the choice of system components. In addition, the need to separate the domain-dependent knowledge to be acquired for each new database from the domain-independent parts of the system influenced the design of the particular data structures (or “knowledge sources”) selected for encoding the information used by these components.

Figure 2 illustrates the major processes of `TEAM`, the various sources of knowledge they use, and the flow of language-processing tasks from the analysis of an English sentence to the generation of a database query. The rectangular boxes represent the processes, and the ovals to their right, the various knowledge sources. The acquisition box on the right points to those knowledge sources that are augmented through interaction with the `DBE`. All other modules and knowledge sources are built into `TEAM` and remain unchanged during acquisition.

In this section we will look at the `TEAM` system from several angles. To begin, we will sketch the overall flow of processing during question-answering, describing the various processes involved in transforming an English query into a formal database query. Because the particular *logical form* (LF) `TEAM` uses to encode the meaning of a query plays a crucial role in mediating between the way queries are posed and the way information is obtained from the database, it affects the design of several components of the system. We then look in somewhat more detail at the data structures that encode domain-specific information. Finally, we discuss the overall strategy used for acquiring information about specific domains and databases.

2.1 Flow of Control

The flow of control during `TEAM`'s translation of a natural-language query into a formal query to the database is illustrated as the path on the left side of Figure 2, from top to bottom. The transformation takes place in two major steps: first, a representation of the literal meaning of the query, or *logical form*, is constructed; second, this logical form is transformed into a database query.

The translation into logical form is performed by the `DIALOGIC` system, which comprises the following components, shown surrounded by the dotted box in Figure 2: the `DIAMOND` parser, the `DIAGRAM` grammar, the lexicon, semantic-interpretation functions, basic pragmatic functions, and procedures for determining the scope of quantifiers.

Since a description of `DIALOGIC` is provided elsewhere [Gros82], let us discuss here only those aspects of the system that were influenced by the development of `TEAM`. Two central data structures in `DIALOGIC` that are affected by `TEAM`'s acquisition process are described: the *lexicon* and

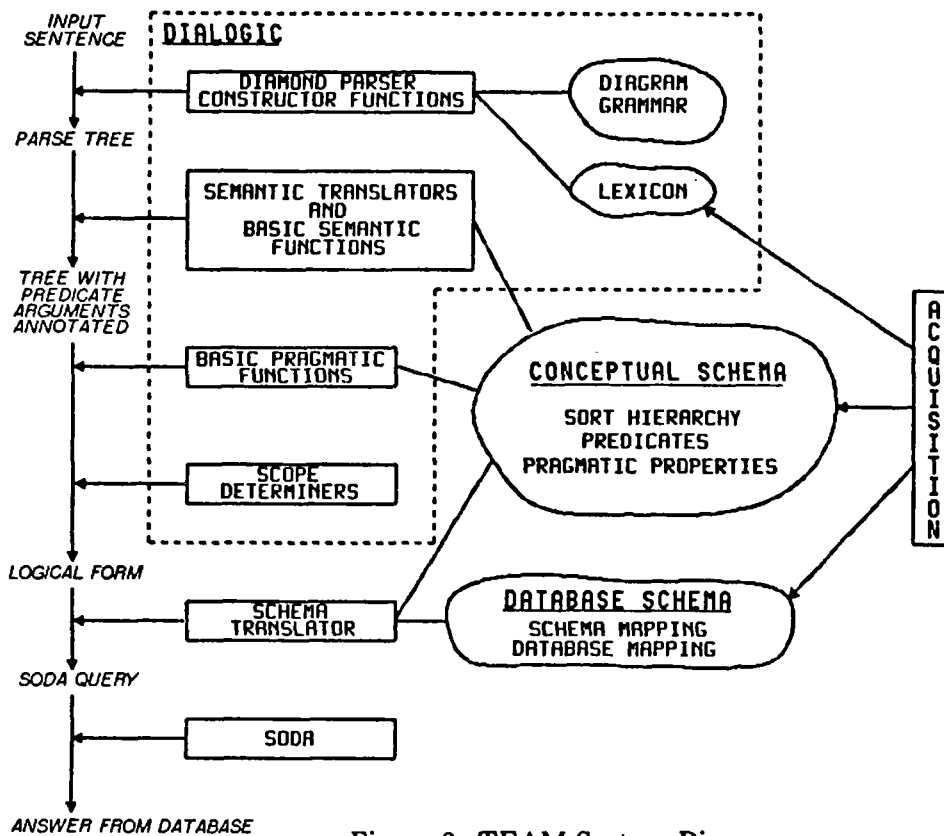


Figure 2: TEAM System Diagram

the *conceptual schema*. To understand the semantic and pragmatic components of TEAM, it is also necessary to appreciate DIALOGIC's separation of semantic interpretation operations into two main classes: *translators*, which define how the interpretations of the constituents of a phrase are combined into the phrase's interpretation; *basic semantic functions*, which are called by the translators to assemble the actual logical-form fragments that form the interpretations of phrases.

In brief, when the end user asks a query, DIALOGIC parses the sentence, producing one or more trees representing possible syntactic structures. The "best" parse tree, based on a priori syntactic criteria, is selected and annotated with semantic information [Robi82, Mart83]. Next, *pragmatic analysis* is applied to assign specific meanings that are relevant to the current domain to noun-noun combinations and to "vague" predicates like HAVE and OF.⁴ Finally, the quantifier-scope determination process, after considering all possible alternatives, determines the best relative scope for the quantifiers in the query. The logical form thus constructed, using a set of predicates that are meaningful with respect to the given domain and database, constitutes an unambiguous representation of the English query.

The logical form produced by DIALOGIC is translated into a query in the SODA [Moor79]⁵ database query language by the *schema translator*. In addition to the conceptual schema, the schema translator uses a database schema that furnishes information about the particular database structures. This schema, described briefly below, is also affected by the acquisition process.

⁴We consider these predicates vague because they can be applied to many kinds of entities; they are replaced by "real" predicates during pragmatic processing.

⁵SODA is actually a query compiler that takes queries in a standard relational formalism and compiles them into optimized queries in the languages of other database management systems; both relational and codicil DBMSs have been accommodated. For our experiments, an interpreter that follows SODA commands to access a small database in primary memory was used in lieu of the actual SODA system.

Finally, the database query produced by the schema translator is given to SODA, which executes the query and displays the answer for the user. SODA was not developed as part of TEAM but was chosen for its features, which are consistent with the overall goal of transportability. SODA was designed for querying distributed databases and is capable of interfacing with several actual database management systems.

The processes TEAM executes in replying to an end user's query are similar to those that any custom-designed NLI would execute. What is different in the case of TEAM is that the modules must be carefully designed to allow for maximal generality, which precludes many of the shortcuts that are common in custom-built NLI systems (e.g., LADDER [Hend77], PLANES [Walt75]). Two techniques that are ruled out are the using a *semantic grammar* and combining the determination of what a query means with the formulation of the DBMS query.

Semantic grammars are based on constituent categories that are chosen not for their ability to embody linguistic generalizations, but rather for the ease of parsing and interpretation that results when the grammar reflects the conceptual structure of the database domain. For example, instead of the general categories of "noun" and "verb phrase," semantic grammars may have categories such as "country" and "location specification." Such grammars are hopelessly tied to a single domain, and probably to a single database as well.

Efficiency also results from mapping a natural-language query directly into the code required for retrieving an answer from the database, but at the cost of being tied to a particular database. A number of database query systems (e.g., LADDER) construct a query directly while parsing the input with semantic grammar rules, but without building any other representation of what the query means.

Although the SODA query that results from the analysis of an English query represents, at least in some sense, the intended meaning of the latter, it does so in a way that directly reflects the structure of the database being queried. Consequently, if two databases encode the same information in different structures, the result will be two different database queries for the same English sentence. For example, if a user asks "How many Swiss mountains are there?" the database queries generated in response to his query can look very different, depending on whether the tuples representing Swiss peaks are distinguished from those representing other peaks by their membership in a different relation, or by the presence of the word "SWZ" in a COUNTRY field.

The problem this creates is not just an aesthetic one: to acquire the semantic and pragmatic rules necessary for generating a database query directly from an English query, TEAM would have to ask the DBE about far more than the structure and contents of the database. Answering the essential questions for such an acquisition would require the kind of expertise in natural-language processing that TEAM is intended to render unnecessary. Thus, the demands of transportability preclude use of the SODA language as the primary representation of the meaning of queries.⁶

2.2 Logical Form

Logical form plays a central role in TEAM: it mediates between the way an end user thinks about the information in a database, as revealed in his queries to the system, and the way information can be retrieved through queries in a formal database-query language. The predicates and terms in the logical form for a particular query are derived from information in the lexicon and conceptual

⁶In addition, DIALOGIC was designed to be a general language understanding system that can be applied to tasks other than database querying. Therefore, it was undesirable to restrict its application by choosing an unsuitable semantic representation.

schema;⁷, hence, the choice of logical form indirectly affects the design of those components of the system and determines, in part, the information the DBE must supply.

The logical form employed by TEAM is first-order logic extended by certain intensional and higher-order operators and augmented with special quantifiers for definite determiners and interrogative determiners. Much research has been done to devise appropriate logical forms for many kinds of sentences [Moor81], but that investigation lies beyond the scope of this article.

2.3 What Information Is Acquired

2.3.1 The Lexicon

The lexicon is a repository of the information about each word that is necessary for morphological, syntactic, and semantic analysis. There are two classes of lexical items: closed and open. Closed classes (e.g., pronouns, conjunctions, and determiners) contain only a finite, usually small number of lexical items. Typically, these words have complex and specialized grammatical functions, along with [at least some] fixed meanings that are independent of the domain. They are likely to occur with high frequency in queries to almost any database. Open classes (e.g., nouns, verbs, adjectives) are much larger and the meanings of their members tend to vary, depending on the particular database and domain. Therefore, most closed-class words are built into the initial TEAM lexicon, while open-class words are acquired for each domain separately. However, there are a number of open-class words, such as those corresponding to concepts in the initial conceptual schema (see Section 2.3.2) and words for common units of measure (e.g., "meter", "pound"), that are so broadly applicable to so many database domains that they are included in the initial lexicon as well.

Lexical entries include those for the names of file subjects (i.e., the entities about which some relation contains information—e.g., peaks for PEAK, and countries for WORLDC in the sample database illustrated in Figure 1.3), field names, and field values. In addition, the DBE can supply adjectives and verbs, as well as synonyms for words already acquired (see Section 2.4). Associated with every lexical entry is syntactic and semantic information for each of its senses. Syntactic information consists of its primary category (e.g., noun, verb, or adjective), subcategory (e.g., count, unit, or mass for nouns; object types for verbs), and morphology. Semantic information depends on the syntactic category. The entry for each noun includes the sort(s) or individual(s) in the conceptual schema (Section 2.3.2) to which that noun can refer. Entries for adjectives and verbs include the conceptual predicate to which they refer, plus information about how the various syntactic constituents of a sentence map onto arguments of the predicate. Scalar adjectives (e.g., "high") also include an indication of direction on the scale (plus or minus).

2.3.2 Conceptual Schema

The *conceptual schema* contains information about the objects, properties, and relations in the domain of the database. It includes sets of individuals, predicates, constraints on the arguments of predicates, and the information needed for certain pragmatic processing. The informational content is similar to that commonly encoded in semantic networks, but the apparatus used is more eclectic. The conceptual schema consists of a *sort hierarchy* and descriptions of various properties of nonsort predicates.

The sort hierarchy relates certain [monadic] predicates that play a primary role in categorizing individuals. These are called *sort predicates* (represented here in italics as in *PERSON*). TEAM was designed with a considerable amount of this conceptual information built in. Figure 3 illustrates

⁷As noted previously, the specific form depends also on general syntactic, semantic, and pragmatic rules for English that are encoded in the various components of DIALOGIC.

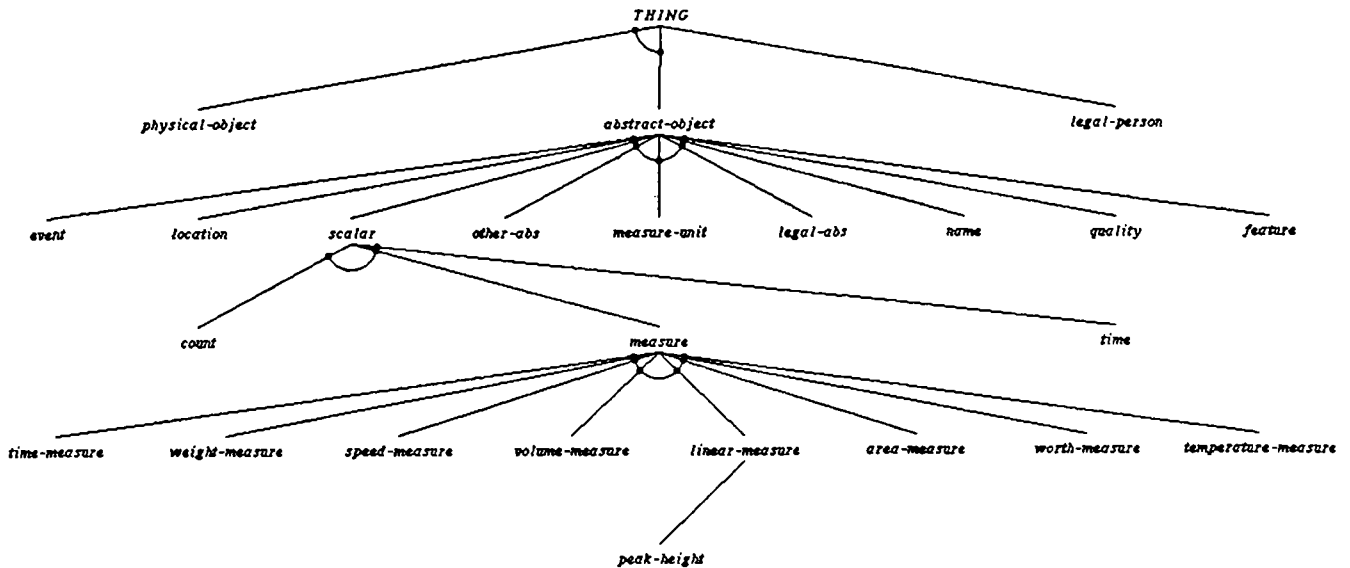


Figure 3: A Fragment of TEAM's Sort Hierarchy

a portion of this hierarchy. Each line connecting levels of the hierarchy signifies a set-subset relationship between two categories of individuals. The sorts connected by the small arcs directly below the nodes are disjoint; that is, no individual can be in two sorts joined in this manner. The sort hierarchy grows as information about a database is acquired. The DBE is required to position some of the newly acquired concepts in their appropriate places in the hierarchy.

Each field in the database is associated with the sort of objects that can appear in that field. Several additional properties are associated with the sorts derived from symbolic fields and from certain kinds of arithmetic fields.

With each sort obtained from a symbolic field, TEAM associates a predicate that encodes the relationship between that sort and the sort of the file subject. For example, for the relation *WORLDC* in Section 1.3, which includes information about capitals and continents, the system would link the sort *WORLDC-CAPITAL* with the predicate **WORLDC-CAPITAL-OF** (in this article, predicates are shown in boldface), which takes two arguments: the first of sort *WORLDC-CAPITAL*, the second of sort *COUNTRY*. This link is used in handling queries like "What is the capital of each country in Europe?" In particular, it is used to determine what it means for a capital to be "of" a country, or for a country to be "in" Europe. Additional properties of the sort indicate whether individual instances of it can modify or stand for instances of the sort of the file subject (e.g., "European countries," but not "Europeans" can be used to refer to the countries *c* satisfying the predication (**CONTINENT-OF** *c* **EUROPE**)).

Sorts that correspond to arithmetic fields containing measures (e.g., length, age) also include information about both the implicit unit of measurement (e.g., feet, years), and the kind of thing being measured (e.g., linear extent, temporal extent).

Several other kinds of information are associated with nonsort predicates. A *delineation* specifies the constraints on the sorts for each of a predicate's arguments; multiple delineations are supported but cannot be described in this brief format. Predicates corresponding to comparative-forming adjectives (e.g., "tall") have two additional properties: a link to the predicate that specifies the degree (e.g., **PEAK-HEIGHT** in our example), and an indication of polarity along the scale being measured (e.g., plus for **TALL**, minus for **SHORT**).

2.3.3 Associated Processes

Several general predicates have semantic and pragmatic specialists associated with them. The semantic specialists are Is-semantic and Degree-semantic; the pragmatic specialists are the Genitive, Noun-noun, Have, Of, General-preposition, Time, Location, Do-specialist, and Comparative.

The Is-semantic specialist is associated with the predicate **IS** and propagates sort restrictions across all the variables that are being equated by the **IS** assertion. This specialist is invoked prior to pragmatic processing (hence the “semantic” label); it attempts to reconcile any conflicts it detects and may revise some sort predications on variables in the process. For example, it is used in processing the query, “What is the area of Nepal?” to ascertain that the variable corresponding to the “what” is a *WORLDC-AREA*, not a *CONT-AREA*.

The Degree-semantic specialist replaces the general predicate **DEGREE-OF** with a more specific one. For example, by determining that predication (**DEGREE-OF** *peak1 x*) refers to the predicate **PEAK-HEIGHT**—i.e., that it is equivalent to the predication (**PEAK-HEIGHT-OF** *peak1 x*)—the specialist allows **TEAM** to further constrain the sort of *x* to be a *linear-measure*, thus allowing the comparative specialist invoked during pragmatic processing to make the right choice between the alternatives of comparing the heights of two objects and comparing an object’s height with a height value.

The Genitive, Noun-noun, Have, and Of specialists replace the vague predicates **GENITIVE**, **NN** (for noun-noun combinations), **HAVE**, and **OF** with more specific ones. The individual specialists differ only slightly, the differences reflecting the special restrictions associated with each construction.

The General-preposition specialist is associated with **ON**, **FROM**, **WITH**, and **IN**, converting these predicates into their appropriate domain-specific counterparts. For example, the Do-specialist determines that the phrase “countries in Asia” means those countries *c* for which the predication (**WORLDC-CONTINENT-OF** *c ASIA*) holds.

The Time-specialist and Location-specialist serve to map **TIME-OF** and **LOCATION-OF** into predicates that are appropriate for the database at hand. They can be invoked obliquely by the interrogative constructions “when” and “where.”

The Do-specialist replaces the predicate **DO** (from the verb “do”) with a more specific verb chosen from those acquired for a domain. Although “do” does not appear as the main verb very often in the database query task, the translators deduce its implied presence in some queries—for instance in such comparative questions as “What countries cover more area than Peru [does]?”.

The comparative specialist examines the two arguments of a comparison to determine whether the comparison to be made is between two attribute values (e.g., Jack’s height and seven feet) or between an entity and some value (e.g., Jack and seven feet). In the latter case, **TEAM** tries to identify the appropriate attribute of the entity (e.g., Jack’s height).

2.3.4 Database Schema

The translation from logical form to SODA query requires knowing the exact structure of the target database and the manner in which the predicates appearing in the logical form are associated with the relations in the database. This information is provided by the *database schema*, which includes the following information⁸:

- Definition of sorts in terms of database relations (subject) or fields (and field value for sorts derived from feature fields).

⁸The schema translator also uses certain information in the conceptual schema, including taxonomic information in the sort hierarchy and delineation information associated with nonsort predicates.

File Menu	WORLDC	BCITY	CONT
PEAK			
Field Menu	BCITY-COUNTRY	BCITY-NAME	BCITY-POP
	CONT-HEMI	CONT-NAME	CONT-POP
	PEAK-HEIGHT	PEAK-NAME	PEAK-VOL
	WORLDC-CAPITAL	WORLDC-CONTINENT	WORLDC-NAME
			CONT-AREA
			PEAK-COUNTRY
			WORLDC-AREA
			WORLDC-POP
Word Menu			
AREA (n)	CAPITAL (n)	CITY (n)	
CONTINENT (n)	COUNTRY (n)	HEIGHT (n)	
HEMI (n)	HEMISPHERE (n)	HIGH (adj)	
LARGE (adj)	LOW (adj)	N (n)	
NAME (n)	NORTHERN (adj)	PEAK (n)	
POP (n)	POPULATION (n)	POPULOUS (adj)	
S (n)	SHORT (adj)	SMALL (adj)	
Question Answering Area			
Field PEAK-HEIGHT is part of an ACTUAL relation.			
Type of field - SYMBOLIC ARITHMETIC FEATURE			
Value type - DATES MEASURES COUNTS			
Are the units implicit? YES NO			
Enter implicit unit - FOOT			
Measure type of this unit - TIME WEIGHT SPEED VOLUME LINEAR AREA WORTH TEMPERATURE OTHER			
Abbreviation for this unit? - FT			
Conversion formula from METERS to FEET - (/ X 0.3048)			
Conversion formula from FEET to METERS - (* X 0.3048)			
Positive adjectives - HIGH TALL			
Negative adjectives - SHORT LOW			

Figure 4: The Acquisition Menu

- List of convenient identifying fields for each sort corresponding to a file subject or field.
- Definition of predicates in terms of actual database relations and attributes; this is done for predicates derived from both actual and virtual relations (for relation subjects and attributes).
- List of each relation's key fields.

The database schema relates all the predicates in the conceptual schema to their representation in a particular database. For each predicate, the database schema generates a logic formula defining the predicate in terms of database relations. For example, the predicate **WORLDC-CAPITAL-OF** has as its associated database schema a formula representing the fact that its first argument is taken from the **WORLDC-CAPITAL** field of a tuple of the **WORLDC** relation, and that its second argument comes from the **WORLDC-NAME** field of the same relation. If a predicate has multiple delineations—i.e., if it applies to different sorts of arguments (e.g., a **HEMISPHERE-OF** predicate could apply to both **COUNTRIES** and **CONTINENTS**)—the schema will include a separate definition for each set of arguments. In some cases (e.g., predicates resulting from the acquisition of some verbs and adjectives), the mapping associated with a predicate indicates that it is equivalent to another [conceptual schema] predicate with certain arguments set to fixed values.

2.4 Acquisition

The acquisition component of **TEAM** is crucial to its success as a transportable system. Recall that one constraint on **TEAM** is that the **DBE** not be required to have any knowledge of **TEAM**'s internal workings, nor about the intricacies of the grammar, nor of computational linguistics in general. Yet detailed information, often necessarily linguistic in its orientation, must somehow be extracted from the **DBE** during an acquisition session. Furthermore, it is desirable that the acquisition component be designed to allow a **DBE** to change answers to questions and add information as he gains experience with **TEAM** and the types of questions that are asked by the end users.

In an attempt to satisfy all these constraints, the menu-oriented system depicted in Figure 4 was developed. The acquisition system consists of a menu of general commands at the very top, three menus associated with relations, fields, and lexical items respectively, and, at the bottom, a

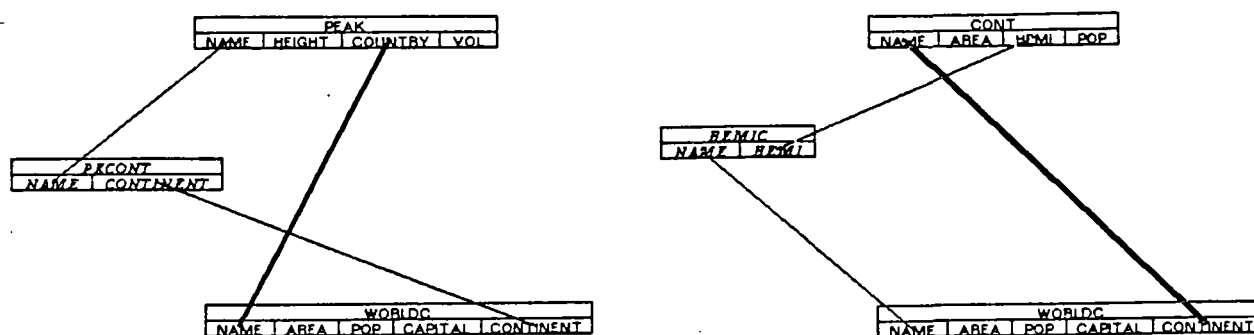


Figure 5: Acquiring the Virtual Relations PKCONT and HEMIC

window for questions and answers. When the DBE uses the mouse to select one of the items from the three menus, a set of questions appears in the question-answering area at the bottom of the display, to which he can then respond.

One of the general principles of acquisition is evident from this display, namely, that the acquisition is centered upon the relations and fields in the database, because this is the information most familiar to the DBE. The answers to each question can affect the lexicon, the conceptual schema, and the database schema. The DBE need not be aware of exactly why TEAM poses the questions it does—all he has to do is answer them correctly. Even the entries displayed in the word menu owe their presence to questions about the database. The DBE volunteers entries to this menu only in the case of verb acquisition, to supply an adjective corresponding to some noun already in TEAM's lexicon, or to enter a synonym for some lexicon-resident word.

The DBE is assumed not to have any knowledge of formal linguistics or of natural-language processing methods. He is assumed, however, to know some general facts about English—for example, what proper nouns, verbs, plurals, and tense are, but nothing more detailed than that. If more sophisticated linguistic information is required, as in the case of verb acquisition, TEAM proceeds by asking questions about sample sentences, allowing the DBE to rely on his intuition as a native speaker, and extracting the information it needs from his responses.

Virtual relations are specified iconically. The left side of Figure 5 shows the acquisition of a virtual relation that identifies the continent (PKCONT-CONTINENT, derived from WORLDC-CONTINENT) of a peak (PKCONT-NAME, from PEAK-NAME) by performing a database join on the PEAK-COUNTRY and WORLDC-CONTINENT fields. Similarly, the right side of Figure 5 shows the acquisition of the virtual relation that encodes the hemisphere (HEMIC-HEMI) of a country (HEMIC-NAME) by joining on the WORLDC-CONTINENT and CONT-NAME fields.

If he wishes, the DBE can change previous answers. Incremental updates are possible because most of the methods for updating the various TEAM structures (lexicon, schemata) were devised to undo the effects of previous answers before the effects of new answers could be asserted. Help information is always available to assist the DBE when he is unsure how to answer a question. Selecting the question text with the mouse produces a more elaborate description of the information TEAM is trying to elicit, usually accompanied by pertinent examples.

Finally, the acquisition component keeps track of what information remains to be supplied before TEAM has the minimum it needs to handle queries. The DBE does not have to determine himself how much information is sufficient; all he has to do is to perceive that no acquisition window indicates remaining unanswered questions. Of course, the DBE can always provide information beyond the minimum—for example, by supplying additional verbs, derived adjectives, or synonyms.

3 Conclusions

TEAM has been tested in a variety of multfile database domains by a fairly large number of people in addition to its original implementation team. While the testing has been much less rigorous than would be required for an actual product, enough has been learned to conclude that the basic ideas “work”—namely, that it is possible to build a natural-language interface that is general enough to allow its adaptation to new domains by users who are familiar with these domains, but are themselves neither experts on the system itself nor specialists in AI or linguistics.

TEAM handles a wide range of verbs, a capability that is absolutely essential for fluent natural-language communication. As it embodies no discourse model, its handling of pronoun resolution and determiner scoping is correspondingly limited. While its grammar coverage is quite extensive, the formalism used to represent it and the processes used to implement it are yielding to newer and more perspicuous designs[Shie84]. We are now investigating ways to provide transportability in natural-language systems that can interact with a variety of software services beyond database access and which more extensive discourse capabilities will be embodied.

Acknowledgments

Jerry R. Hobbs, Robert C. Moore, Jane J. Robinson, and Daniel Sagalowicz played important roles in the design of TEAM. Armar Archbold, Norman Haas, Gary Hendrix, Lorna Shinkle, Mark Stickel and David H. Warren also contributed to the project.⁹

References

- [Gros85] Barbara Grosz, Douglas E. Appelt, Paul Martin, and Fernando Pereira. *TEAM: An Experiment in the Design of Transportable Natural Language Interfaces*. Technical Note, Artificial Intelligence Center, SRI International, Menlo Park, California, 1985.
- [Gros82] Barbara Grosz, Norman Haas, Gary G. Hendrix, Jerry Hobbs, Paul Martin, Robert Moore, Jane Robinson, and Stan Rosenschein. *DIALOGIC: A Core Natural-Language Processing System*. Technical Note 270, Artificial Intelligence Center, SRI International, Menlo Park, California, November 1982.
- [Hend77] Gary G. Hendrix. Human engineering for applied natural language processing. In *Proc. of the Fifth International Joint Conference on Artificial Intelligence*, pages 183–191, International Joint Conferences on Artificial Intelligence, Cambridge, Massachusetts, August 1977.
- [Mart83] Paul Martin, Douglas Appelt, and Fernando Pereira. Transportability and generality in a natural-language interface system. In Alan Bundy, editor, *Proc. of the Eight International Joint Conference on Artificial Intelligence*, pages 573–581, International Joint Conferences on Artificial Intelligence, August 1983.
- [Moor79] Robert C. Moore. *Handling Complex Queries in a Distributed Database*. Technical Note-170, Artificial Intelligence Center, SRI International, Menlo Park, California, October 1979.
- [Moor81] Robert C. Moore. Problems in logical form. In *Proc. of the 19th Annual Meeting of the Association for Computational Linguistics*, Stanford, California, 1981.

⁹The development of TEAM was supported by DARPA contracts N00039-80-C-0645, N00039-83-C-0109, and N00039-80-C-0575; the National Library of Medicine NIH grant LM03611; and NSF grant IST-8209346.

- [Robi82] Jane J. Robinson. Diagram: a grammar for dialogues. *Communications of the ACM*, 25(1):27-47, 1982.
- [Shie84] Stuart M. Shieber. The design of a computer language for linguistic information. In *Proc. of Coling84*, pages 362-366, Association for Computational Linguistics, June 1984.
- [Walt75] David Waltz. Natural-language access to a large data base: an engineering approach. In *Proc. of the Fourth International Joint Conference on Artificial Intelligence*, pages 868-872, International Conferences on Artificial Intelligence, September 1975.

A MULTILINGUAL INTERFACE TO DATABASES

Hubert Lehmann, Nikolaus Ott, Magdalena Zoeppritz
IBM Germany, Heidelberg Scientific Center

Abstract

The User Specialty Languages (USL) System, a portable interface to relational databases in restricted English, French, German, Italian, and Spanish is described. We briefly discuss our design objectives, theoretical and practical problems we encountered during system realization, and the consequences we have drawn for a successor project. The German and English versions of the USL System have been extensively evaluated with real users and real applications, which not only showed us where we could improve our system but also provided valuable insights for the methods of software ergonomics.

Introduction

When we talk about interaction with databases we must clarify two things: 1. who are the groups of people who want to obtain information, and 2. what are the operations to be performed on the database to yield the information desired? Then we can think about *how* these operations are to be specified by a given user.

A number of *query languages* have been developed during the 70's and efforts to show their "user-friendliness", their appropriateness for "non-DP experts" have been made with greater or lesser success (cf. e.g. [LEHM 79] for a survey). A different approach is to regard human question-answering dialog as a model for the interaction with a database, as presumably it is best to talk to the computer in one's own language. The problem then is to relate natural language expressions to data in the database and to the operations to be performed on them.

In the USL project we showed that

- fragments of natural language can be implemented that are large enough to be usable for database access,
- the syntax and semantics of such fragments can be described in such a way that the system becomes independent of the particular domain of discourse (this property has become known as *(trans)portability*),
- adaptation to a new domain can be achieved without training in linguistics,
- natural language interfaces can be built which operate on standard databases (i.e. neither require special representation nor manipulation of data).

Design principles

The USL System was designed with the objectives to be usable in realistic applications, to be portable, to enable adaptation to new domains by non-linguists, and to provide an interface to standard databases. A later goal was the adaptation to a variety of different languages, which brought in a few new aspects, but was on the whole a relatively straightforward task.

These objectives had a number of consequences for the design of the USL system which we discuss in the following sections.

Consequences of portability

A system is portable if it is "easy" to adapt it to new domains of discourse. We do not believe it is possible to give a proper definition of portability, but we can find a number of features that we feel are important:

1. the distinction of domain dependent and domain independent words (e.g. the meaning of prepositions should be invariant),
2. domain independent syntax (where a syntactic category "ship" as in versions of Semantic Grammar is not allowed),
3. the interpretation of syntactic constructions independent of the domain of discourse (e.g. what does it mean for a noun to be modified by a relative clause),
4. separation of domain dependent information from domain independent information (adaptation to a new domain should not involve programming)

Application development by non-linguists

Once a system has the features mentioned above for portability, a big step towards application development by non-linguists has been made. What has to be done in addition is to find ways to elicit the required domain dependent information from application developers. In the design of the USL System we chose to restrict the linguistic information we use for domain specific words to an absolute minimum, and to write a vocabulary definition program that elicits all required morphological, syntactic, and semantic information from the user by giving appropriate examples and guidelines where needed.

Linguistic coverage

Ideally the linguistic coverage of a natural language system is such that a user never falls outside its boundaries. Realistically, it is a com-

promise between what is feasible with the current state of computational linguistics and what is necessary to perform the desired database operations. To be acceptable the compromise must be such that a) there always is *some* (easily identifiable) way for the user to get his result, b) when a linguistic construction is understood in one context, it should be understood in all appropriate contexts.

Since there is no way to know in advance how these criteria can be fulfilled we tried to find out with user studies whether the linguistic coverage provided in the USL System is in fact acceptable. The linguistic coverage includes interrogative, imperative, and declarative sentences. The following constructions are provided:

- nouns and noun phrases (definiteness, quantification, interrogative pronouns, personal pronouns, possessive pronouns, relative pronouns),
- verbs (including "to be" and "to have"),
- adjectives (including gradation, modification by modal adverbial and by ordinal number),
- temporal adverbials,
- units of measure,
- comparatives,
- verb complements (subjects and nominative complements, accusative objects, dative objects, genitive objects, prepositional objects),
- noun complements (relative clauses, participial attribute phrases, genitive attributes, appositions, prepositional attributes),
- complements of noun and verb (negation, adverbials of place and time)
- coordination for nouns, noun phrases, adjectives, verb complexes and sentences
- computational expressions (aggregate functions, arithmetic expressions)

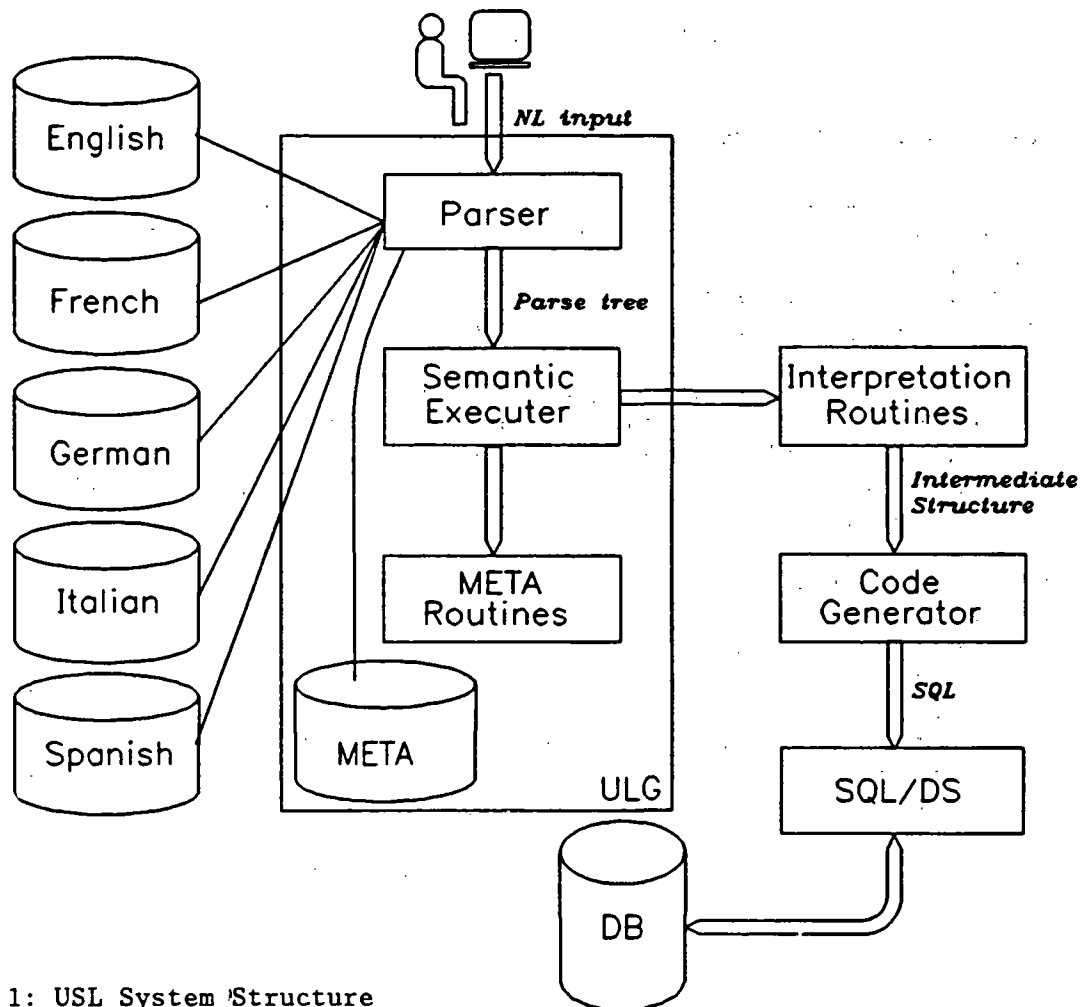


Fig. 1: USL System Structure

System structure

In this section we give an overview of the structure of the USL System (cf. fig. 1) which consists of

1. the language processing component ULG [IBM 81] which includes a parser, a semantic executer, and the language META for describing grammars,
2. grammars for French, English, German, Italian, and Spanish,
3. a set of about 75 interpretation routines to transform parse trees into Intermediate Structures,
4. a code generator to generate expressions in the query language SQL from Intermediate Structures,
5. a high-level optimizer for SQL queries,
6. the relational database system SQL/DS [IBM 83],
7. a vocabulary definition facility.

In the following sections we will give some details on these system components.

Syntax analysis

The parser used in the USL system was originally described by Kay [KAY 67] and subsequently implemented in the REL system [THOM 69]. For USL we adopted a modified version of this parser which is part of the ULG system due to Bertrand and Daudenarde (published as [IBM 81]) which has the following characteristics:

- it accepts general phrase structure grammars written in META, which is similar in appearance to Backus-Naur-Form;
- with any rule it allows the invocation of arbitrary routines to control its application or to perform arbitrary actions
- it allows sophisticated checking and setting of syntactic features;
- it produces all parses in parallel;
- it operates in a bottom-up fashion, working from right to left.

The ULG parser together with the META grammar writing formalism is a powerful tool for grammatical description. It is easy to express in it feature operations as suggested by Gazdar with his Generalized Phrase Structure Grammar (GPSG) or in Lexical Functional Grammar (LFG). The only linguistic tools missing are transformational cycles in the sense of Transformational Grammar. For this reason, necessary rearrangement and reconstruction is done by means of interpretation routines.

The German grammar of the USL System is described in detail by Zoeppritz in [ZOEPE 84a]. It has the coverage outlined above and the most thoroughly tested grammar we have. It comprises about 700 rules and thus represents one of the most comprehensive grammars operational in a computer. Compared to English, German is interesting for its inflectional system, its relatively free word order, and its disjoint constituents (e.g. separable verb prefixes). These aspects are responsible for the much larger size of the German grammar compared to our English grammar which has about 450 rules, but approximately the same linguistic coverage.

A Spanish grammar for USL was written and documented by Sopena [SOPE 82], a French grammar by Roesler [ROES 85]. The documentation of the Italian grammar which has also been completed, is forthcoming.

In addition to syntactic structures the grammars contain sets of about 450 structural words (domain independent words) such as 'the', 'of', 'have', 'be'.

which organizations do france and spain belong to

Parse tree

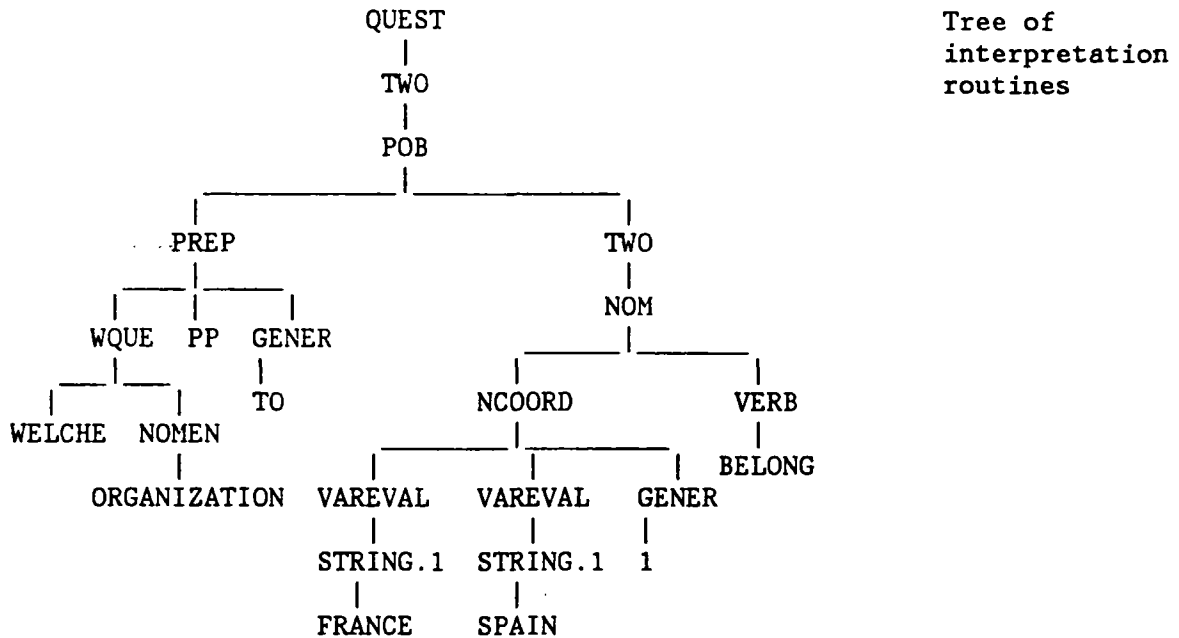
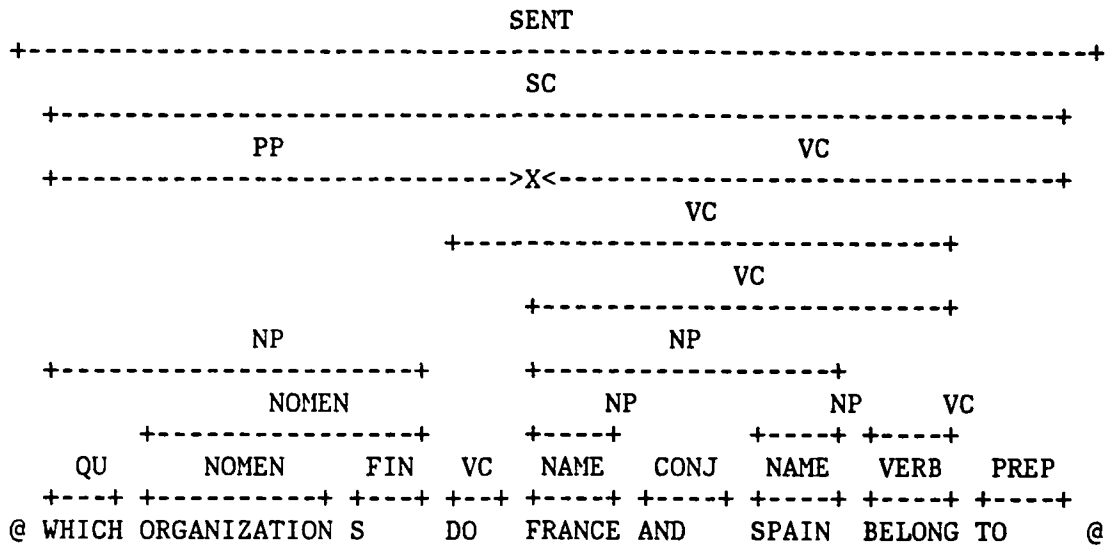


Fig. 2.1: Syntax analysis

Parsing yields one or more trees spanning the input sentence. Each node in the tree is associated with the name of the interpretation routine that was specified in the grammar rule according to which the node has been constructed (cf. the example in fig. 2.1).

Interpretation

In this section we give a brief summary of how interpretation of natural language expressions is done in the USL System. A more detailed description can be found in [LEHM 78] and in [GUEN 84].

We take the view that a database is a model of the world which consists of objects and relationships. Words express concepts which are in some way, namely by giving their extensions, represented in the database. Concepts are represented then as values or relations: common nouns, adjectives, and verbs are associated with relations, proper names with values in the database. Thus a word like *population* may be associated with a set of pairs of population figures and country names, and a query like *what is Spain's population* must then be translated into an SQL query referring to this set. An important question to decide is what columns of a relation should correspond to what parts of a sentence. This was solved by the introduction of role names which relate complements of verbs, nouns, and adjectives to columns of relations.

Interpretation of linguistic constructions is realized by systematic, i.e. domain independent, translation into SQL expressions which are evaluated by the database system. SQL expressions are not directly generated from parse trees, as it would be difficult to correctly interpret the scope of quantifiers in this way or to cope with a number of other linguistic phenomena. Therefore we chose to transform parse trees into so-called Intermediate Structures, which are trees quite similar to F-structures in LFG (but developed independently). This transformation is the result of the execution of the interpretation routines mentioned above. In a sense the interpretation routines express the meaning of word classes and syntactic constructions. There are routines attaching a relative clause to a noun, modifying nouns by interrogatives, etc. Thus all languages having relative clauses or interrogatives modifying nouns can be described using these routines, regardless of the particular syntactic form in which they are realized.

which organizations do france and spain belong to

R: BELONG Intermediate
structure
A(TO):1 1: R: ORGANIZATION
A(NOM):1 2: K: 1
 'FRANCE'
 'SPAIN'

Additional fields contain information on negation,
quantifiers, possessives and their referents,
definiteness, interrogatives, comparatives, etc.

UNOPTIMIZED QUERY: After recursive
code generation:
query against views

```
SELECT DISTINCT X03.WNOM_ORGANIZATION
FROM BELONG X02,BELONG X01,ORGANIZATION X03
WHERE (((X02.WNOM_COUNTRY = 'FRANCE'
AND X01.WNOM_COUNTRY = 'SPAIN')
AND X01.WTO_ORGANIZATION = X02.WTO_ORGANIZATION)
AND X01.WTO_ORGANIZATION = X03.WNOM_ORGANIZATION)
```

JOIN OF X03B1 AND X02B1 REMOVED BY ABSORPTION PROPERTY

OPTIMIZED QUERY: Optimized query
sent to data base

```
SELECT DISTINCT X02B1.ORGANIZATION
FROM OTT.BELONGB X02B1,OTT.BELONGB X01B1
WHERE X02B1.COUNTRY = 'FRANCE'
AND X01B1.COUNTRY = 'SPAIN'
AND X01B1.ORGANIZATION = X02B1.ORGANIZATION
```

ORGANIZATION Answer returned

CCD
ECE
FAO
⋮

Fig. 2.2: Code generation and optimization

The interpretation routines are invoked by the semantic executer in the order in which they appear in the parse tree, starting at its root. (The results of this and the following steps are shown in fig. 2.2.)

The code generation program is called after the completion of the Intermediate Structure which is then processed recursively, and thus gradually the executable query is built up. Quantification, negation, and coordination may lead to more than one SQL query being generated from a given Intermediate Structure.

The resulting SQL query or queries are transmitted to an optimizer which eliminates redundant join operations that appear for instance when different views are defined on the same base relation (cf. [OTT 85] for details). The resulting SQL query is sent to SQL/DS for evaluation. In case of yes/no questions the answer YES or NO is displayed, complement questions are answered by one or (in some cases of coordination) several tables.

Evaluation of usability

Few natural language systems have reached a point where it made sense to evaluate their usability. Evaluation methods for software in general are still not very well developed, and evaluating natural language systems seems to be particularly difficult. The USL System was evaluated both in laboratory and "field" studies. Laboratory studies provide good control of the environment, but there is always a danger that the wrong hypotheses are tested (cf. [ZOEP 84b]). The USL System was compared to SQL and it was found that for very simple and very complex queries SQL was at some advantage whereas for medium complexity queries USL was superior. Further it was found that the restrictions of USL had to be learned, but also that they could be learned (for details see [JARK 85a] and [VASS 83]). An extended case study was very successfully conducted with 3 school teachers who posed more than 7000 queries with a total error rate of about 8 percent (details are presented in [KRAU 80] and [KRAU 82]). A number of smaller studies with different application domains were also successfully conducted, which in our view confirms our claim that the USL System is both a portable and a usable interface to a database system. (Cf. also [JARK 85b] in this issue.)

Conclusions

The construction and evaluation of the USL System has been an important step towards viable natural language interfaces to databases. Yet the experience with the USL System also showed where further research is desirable. We want to point out three areas:

1. While the fragments of natural language provided in the USL System proved to be usable, improved facilities for meta-communication are needed: This concerns primarily error situations (parsing ill-formed input, constructions outside the scope of the system), inappropriate questions (e.g. violated presuppositions), possibilities to review the capabilities of the system and the contents of the database.
2. Grammars should be extended to virtually complete coverage even if constructions cannot be properly interpreted: This is important to provide better help to users when they move outside the system's linguistic capabilities.
3. Modelling of complex domains needs further investigation, even when portability has been achieved: This typically involves mass terms, complex spatio-temporal relations, and a number of concepts whose relational representation involves relation-valued attributes.

In conclusion we want to mention that the USL system forms the basis for the natural language component of an expert system project we are currently working on and which has been described in [GUEN 84]. It is the goal of this project (called Linguistics and Logic based Legal Expert System) to investigate problems of discourse representation and processing of knowledge in the domain of German traffic law. Emphasis is laid on the integration of expert system technology and natural language processing methods in order to facilitate knowledge acquisition and problem solving through natural language discourse.

References

[GUEN 84] F. Guenther, H. Lehmann, "Automatic Generation of Discourse Representation Structures", *Proc. COLING'84*

[IBM 81] IBM, *User Language Generator: Program Description / Operations Manual*, IBM France, Paris.

[IBM 83] IBM, *SQL/Data System General Information*, GH24-5013, IBM Corp., Endicott, USA.

[JARK 85a] M. Jarke, J. Krause, Y. Vassiliou, "Studies in the Evaluation of a Domain-Independent Natural Language Query System", in L. Bolc (ed.), *Cooperative Interactive Information Systems*, Springer, Heidelberg (to appear).

[JARK 85b] (this issue).

[KAY 67] M. Kay, "Experiments with a Powerful Parser", *Second International Conference on Computational Linguistics*, Grenoble, August 1967.

[KRAU 80] J. Krause, "Natural Language Access to Information Systems. An Evaluation Study of its Acceptance by End Users", *Information Systems*, vol. 5, pp. 297 - 318.

[KRAU 82] J. Krause, *Mensch-Maschine Interaktion in natuerlicher Sprache*, Niemeyer, Tuebingen.

[LEHM 78] H. Lehmann, "Interpretation of Natural Language in an Information System", *IBM J. Res. Develop.*, vol. 22, pp. 560 - 572.

[LEHM 79] H. Lehmann, A. Blaser, "Query Languages in Database Systems", in K. H. Boehling, P. P. Spies (eds.): *Proc. of the 9th Annual Meeting of the Gesellschaft fuer Informatik (GI)*, Springer, Heidelberg, p. 64 - 80.

[OTT 85] N. Ott, K. Horlaender, "Removing Redundant Join Operations in Queries Involving Views", *Information Systems*, vol. 10, no. 2 (1985).

[ROES 85] S. Roesler, "Syntax for French in the User Specialty Languages System", *TR 85.04.003*, IBM Heidelberg Scientific Center.

[SOPE 82] L. Sopena-Pastor, "Documentation of the Spanish Grammar", *TR 82.05.004*, IBM Heidelberg Scientific Center.

[THOM 69] F. B. Thompson, P. C. Lockemann, B. H. Dostert, R. S. Deverill, "REL: A Rapidly Extensible Language System", *Proc. 24th National ACM Conf.*, New York, August 1969.

[VASS 83] Y. Vassiliou, M. Jarke, E. A. Stohr, J. A. Turner, N. H. White, "Natural Language for Database Queries: A Laboratory Study", *MIS Quartely*, December 1983, pp. 47 - 61.

[ZOEP 84a] M. Zoeppritz, *Syntax for German in the User Specialty Languages System*, Niemeyer, Tuebingen.

[ZOEP 84b] M. Zoeppritz, "Investigating Human Factors in Natural Language Data Base Query", *TR 84.08.008*, IBM Heidelberg Scientific Center.

EVALUATION AND ASSESSMENT OF A DOMAIN-INDEPENDENT
NATURAL LANGUAGE QUERY SYSTEM

Matthias Jarke (*), Jurgen Krause (**), Yannis Vassiliou (*)

Edward Stohr (*), Jon Turner (*), and Norman White (*)

Abstract

This paper presents a synopsis of the results of several empirical studies which investigated the same domain-independent natural language query system, using various applications in two different natural languages -- English and German. Taken together, these experiments involved about 100 subjects and over 12,000 queries. Discrepancies and open questions requiring additional research are highlighted.

1.0 INTRODUCTION

There is growing consensus that some of the most crucial questions concerning the feasibility and desirability of natural language interfaces to databases can only be resolved by empirical research. Specifically, three central questions concerning NLI themselves are still awaiting an answer.

(1) Can NLI be implemented at all? It seems clear that a full natural language system corresponding to interhuman communication is presently infeasible; any practice-oriented NLI must be application-specific. On the other hand, a NLI would be unacceptable if each user required support by language engineers for an excessive period of time, if the subset of natural language that can be implemented efficiently were not sufficient to support a practical application, or if users had insurmountable difficulties recognizing the boundaries of the implemented subset.

(2) If NLI can be implemented, do they support human problem solving more successfully than competing end user interfaces, such as formal query languages? A meaningful answer to this question requires measurements beyond the percentage of submitted queries that is accepted by a system.

(*) Computer Applications and Information Systems, Graduate School of Business Administration, New York University, 90 Trinity Place, New York, N.Y. 10006, USA

(**) Linguistische Informationswissenschaft, Universitat Regensburg, Universitätsstr. 31, 8400 Regensburg, West Germany

(3) How difficult is it to transport a NLI to a new application? This question is important since it may not be economically feasible to develop a completely new NLI for each new application -- and maybe for each user of each application!

This paper focuses on NLI for database querying (NLQS). Within this group, two essentially different approaches can be distinguished: domain-specific NLQS in which a large portion of the system has to be redeveloped for each new application, and domain-independent systems in which most of the system is portable between applications and the parts to be changed are clearly isolated and relatively small. The latter have also being referred to as "subset" systems - drawing on general language knowledge, application-specific vocabularies, and the database itself.

This paper examines the three questions raised above in the context of a particular restricted subset NLQS, called USL (see, Lehmann et al in this issue), which represents this type of natural language system in a rather pure form. There seems to be no NLQS or other NLI that has been subjected to a comparable number of empirical studies. The first objective of this paper is to present -- in a common framework -- the experience gained from multiple evaluation methods applied to the same system. A second objective is to contribute to a better understanding of the overall feasibility and desirability of the domain-independent approach to NLI, based on the empirical assessment of one specific system.

2.0 RESEARCH OVERVIEW

The NLQS whose evaluation is reported here provides a natural language interface (English, French, German, Italian, and Spanish) to relational databases. The system does not engage the user in clarification dialog, and to that extent the system is similar to any formal database query language. An extended description of the system is given in another article of this issue (see, Lehmann et al).

2.1 Basic Evaluation Methodologies

The simplest and most widely used approach for the evaluation of NLI is the exchange of intuitive arguments about implementation techniques and language features. For example, the information about natural language systems found in the literature is typically highlighted with a list of supported features (e.g., coordination or ellipsis).

Such a list is only useful for the features not included. It can be very misleading since it rarely addresses the important question: "to what degree is the feature supported?" Therefore, it becomes almost impossible to effectively evaluate the usability of any system based on the information given by the system description. Furthermore, opposing arguments of comparable plausibility are confronted without much prospect for a purely argumentative synthesis. Empirical evaluation research may lead out of this dilemma.

Answering the three questions, set forth in the introduction with respect to the domain-independent type of NLI, requires a carefully designed methodology for generating and verifying research questions. In this subsection, some of the basic design parameters for empirical investigations of NLQS will be analyzed. The leftmost two columns of Table 2-1 provide an overview of such parameters (compare also [KRAU82; TURN84]).

DECISION VARIABLE	DESIGN ALTERNATIVES	STAGE A	STAGE B	STAGE C
evaluation team	designers outside researchers	x	x	x
evaluation strategy	absolute comparative	x	x field (x) lab	x
evaluation criteria	quantitative: success effort qualitative: problems strategies level: work task query	x (x) x	x (x) x x	x x (x) x x
evaluation object	simulated NLI real NLI	x	x	(x) x
type of study	laboratory experiment field study	x	(x) x	x x
subject selection	students paid subjects end users, novices end users, experts	x x	x lab x KFG x TA	x
database and application	structure: simple medium complex size: small large	x x x	x x	x x lab x field

TABLE 2-1: Design Parameters for Empirical NLQS Evaluation Studies and Characterization of the Studies Reported in this Paper

Evaluation Team. The first step in evaluating a natural language system empirically is an on-site test of the parser, often termed as an acceptance test. After an iterative process (each iteration corresponding to an improvement of the grammar and the interpretation routines) the system may reach a steady 'acceptable' state.

There is certainly a need for performing this kind of evaluation but there is also the danger of deriving optimistic conclusions about the usability of the system, after attaining a steady state, or of abandoning useful research efforts if a steady state is not reached. Better control is provided by formal evaluation studies conducted by researchers outside the design team. Such an empirical evaluation can be seen as part of a cost-benefit analysis required before introducing a query language into an actual user environment [JARK82]. Several design decisions are of critical importance in this process.

Evaluation Strategy. The first issue is whether the NLI should be evaluated in the absolute or compared to a competing interface, such as a formal query language. Some useful analyses (e.g., of user problem solving strategies) can be performed in the first case. However, performance evaluations using this strategy are meaningful only if the system under study is either close to perfect, or the results are so disastrous that any alternative would be preferable. Otherwise, a comparative study is necessary.

Evaluation Criteria. This discussion leads to the second design question: how can one measure the costs and benefits of a natural language user interface? Of interest are: the success rate of users working with the system, the effort to achieve such success (or failure), the language and system related problems, the strategies users develop to work around the limitations, and finally the subjective perceptions and opinions of the users. Additional criteria may be required to control for confounding outside factors.

Orthogonal to these criteria are the amount of skills the user has acquired [SCHN84], and the level on which performance is evaluated. The former refers to the differentiation between learning and routine task performance [MORA81], which is closely related to the definition of user types [JARK82]. The latter addresses the distinction between the solution of a problem or work task, for which the database is a tool among others employed by the user, and the generation of an answer to a specific database query.

Evaluation Object. The organizational setting of the study must be decided. Some studies assume a simulated rather than a real NLI (e.g., [CHAP73; SMAL77; SHNE80]). Studies of this type can give valuable hints concerning the desirability of NLI but are usually unsuited for establishing their feasibility.

Type of Study. A more important distinction is between laboratory experiments and field studies of real systems. Laboratory experiments allow for a controlled setting. Methodologies to run them have been extensively studied, and the experiments are economically affordable. Such studies, if performed correctly, are best suited for examining the short-term 'learnability' of a language, identifying language constructs likely to cause user difficulties, and for estimating the number and type of words used for a particular set of tasks, as well as the language features most likely to be employed.

On the other hand, drawing practical conclusions about the overall usability of a natural language system from laboratory experiments may be dangerous [REIS81].

Despite the critical remarks by Tennant [1979], the lack of field studies has hardly changed. Aside from the studies described in this paper, the main exception is a year-long field study of TQA, yielding about 700 queries with an acceptance quote of approximately 65% [DAME79]. However, the setting did not allow for the implementation of detailed controls, nor was this intended. Some even more informal studies [HARR77] report only about 20% language-related errors but disregard certain other kinds of failure of the man-machine communication. In general, field studies should be suitable for the evaluation of actual task performance over an extended time period if close observation or carefully designed controls permit the elimination of outside confounding factors.

Subject Selection. The type and intrinsic motivation of users often has a strong impact on the results of laboratory and field studies. The preferred type of users, actual end users, can be quite demanding and may actually abandon system usage if an alternative way to solve their problems is available. On the other hand, student subjects may be less motivated to achieve good performance. The intermediate solution, using paid subjects, may yield good results if their compensation is related to their success with the system or a good motivation can be achieved in a different way.

Database and Application. Last but not least, the size and complexity of both the application domain and the underlying database may influence the outcome of the experiments, by response time effects as well as by the impact of complexity on the user's ability to fully understand the application.

3.0 OVERVIEW OF EVALUATION STUDIES

Experiments with the NLQS have been conducted by different research groups (IBM Scientific Center Heidelberg, University of Regensburg, New York University), using two different natural languages (German and English) and various experimental designs. Three stages of experimentation can be distinguished.

In the first phase (stage A), the development team tested the system informally to uncover errors and gaps in coverage. However, with the exception of one application, no actual field usage was reached since high error rates required continuous drastic changes of the prototype.

The second set of experiments (stage B, the KFG study at Heidelberg and at the University of Regensburg since 1978) was still performed in part at the development site and with technical support by the development team but by an external researcher. At the heart of these experiments was a long term (16 months) observation of a single user working on a practical application. Detailed qualitative analyses were performed, and the original field study was complemented by another field study and several minor laboratory experiments.

The evaluation studies of stage B can be seen as parts of an extended evaluation scheme, outlined in Figure 3-1. The plan starts with a real application to be analyzed in a field study. Laboratory experiments are based on a typical session of this real application. The field studies and laboratory experiments of stage B consisted of three subgroups:

1. A field study with teachers of the Karl-Friedrich-Gymnasium (KFG) at Mannheim in West Germany (the KFG field study).
2. An effort to transport the same system version to another application (the TA field study).
3. Several laboratory tests to compare error rates in the KFG field study with those achieved by using formal query languages.

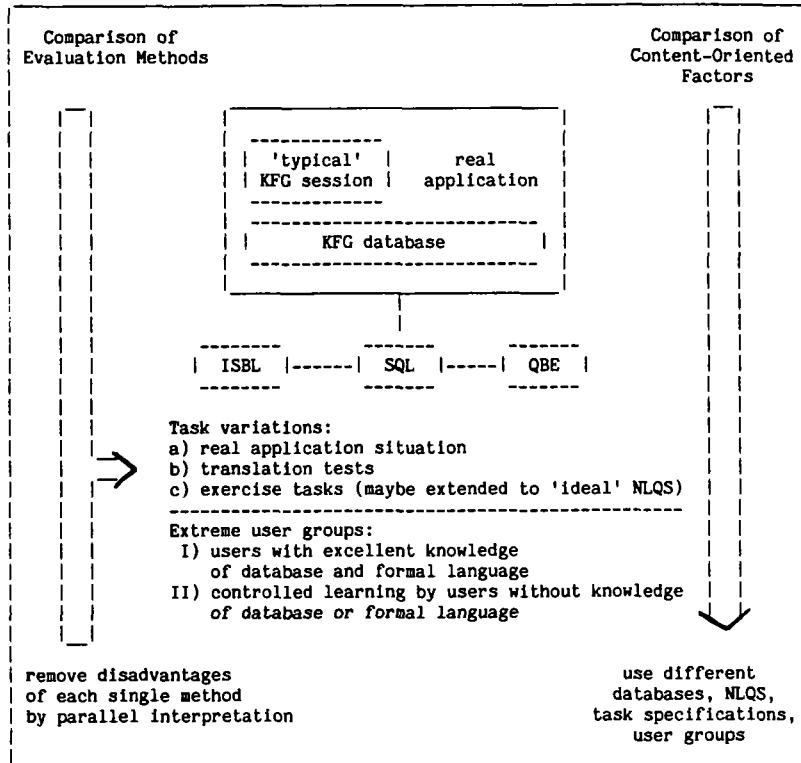


FIGURE 3-1: Evaluation Plan - KFG Studies

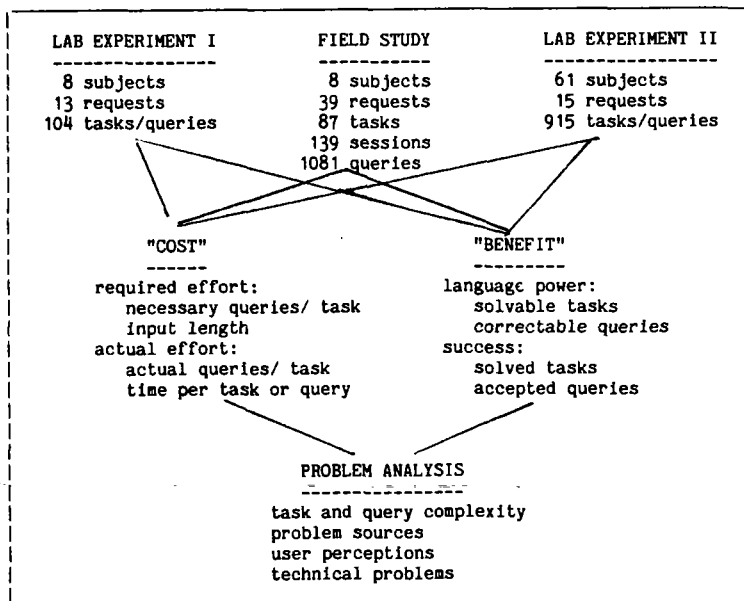


FIGURE 3-2: Evaluation Plan and Descriptive Statistics - ALP Studies

For the third series of experiments (stage C, the Advanced Language Project (ALP) at New York University from 1981-1983), the system was transferred to a different natural language (English), and to a site where little linguistic or technical support by the development team was available. A quantitatively oriented evaluation strategy was chosen for comparing the NLI to a formal database query language in a partially controlled field study and two controlled laboratory experiments. The rightmost columns of Table 3-1 characterize each of the three stages by the design parameters presented in the previous section. Detailed information about each stage can be found in [LEHM78; KRAU80; JARK85a; VASS83] and in [JARK85b]. This paper concentrates on the collective conclusions from all studies.

The purpose of ALP was to study the English language version of the system in a real application (alumni fund raising), and in a location remote from that of the development team. The database contained four base relations with approximately 100,000 tuples, substantially more than in previous applications of the natural language system.

The research centered on the question of whether -- in this setting -- the system (as an example for a transportable NLQS) is superior to a formal query language, such as SQL, in terms of learnability, problem-solving success, or effort to use. A comparative study design and mostly quantitative evaluation criteria were chosen for all experiments.

The project design coupled a field study with two controlled laboratory experiments. The experiments began in 1981 with the design and generation of the database and of the application-specific vocabulary, followed by the application and language training and testing of 8 experimental subjects. This skill acquisition phase was organized as a controlled laboratory experiment. After subjects had reached sufficient proficiency in application and language, they performed real work tasks in the actual setting for more than six months (the field study). The field study raised several additional research questions, and the results of the first laboratory test had to be confirmed with a larger number of subjects. Therefore, a second laboratory experiment with 61 subjects was conducted as a paper-and-pencil test in late 1982. Global design and descriptive statistics of the ALP project are summarized in Figure 3-2.

4.0 SYNOPSIS OF EMPIRICAL RESULTS

In this section, we investigate the relationships between the data gained by the evaluation studies of stage B (KFG) and stage C (ALP). Having a common empirical base, we point out the major results and attempt to explain the differences.

Based on results of both studies, five statements seem to have a fairly strong empirical backing.

1. Users do not communicate with a NLI in the way they do with a human, as suggested in [CHAP73]. In particular, they are very careful in typing input, as evidenced by a low percentage of typographical errors. It is open, how this would change with widespread availability of automatic spelling correction for NLI.

2. Small vocabulary subsets are sufficient for restricted application areas. This result may not extend to some of the knowledge-based systems which require the definition of all words used (including, in particular, values appearing in the database, see, e.g., [BATE83]).
3. Natural language is more concise than formal query languages. In particular, SQL requires substantially longer input even for rather simple queries.
4. Formal query languages cannot be rejected on the grounds that a substantial effort is needed to learn them.
5. Neither study confirmed the fear that natural language queries grow more and more complex over time. Rather, there seems to be evidence that users adapt to what they perceive as the system's limitations. In the KFG field study, query complexity remained about stable over time, whereas in ALP it actually decreased.

USER GROUP	NO. USERS	SESSIONS	QUERIES	ERROR RATE
STAGE A				
Planning	2	7	59	46.0%
School	1	6	356	12.9%
Reception	1	4	115	47.0%
Rooms	3	47	781	39.9%
STAGE B				
KFG main 1	1	39	6603	6.9%
KFG user 2	1	5	582	16.9%
KFG user 3	1	1	93	31.1%
TA study	1	1	67	52.7%
STAGE C (*)				
ALP phase 1	4	34	256	77.0%
w/o line noise				69.1%
ALP phase 2	4	31	271	82.3%
w/o line noise				74.9%

(*) ALP figures do not contain incomplete query typing attempts.

TABLE 4-1: Performance Overview NLQS Field Studies

On first sight, the main discrepancies between the results of ALP and KFG concern the error rates (Table 4-1). These are much higher in ALP than in the KFG studies, yet comparable to the stage A studies. The most plausible explanation regarding the differences in the laboratory experiments seem to be deviations in the test designs. In the field study, a second startling discrepancy is visible in the number of queries per session, resulting from the differences in time per submitted query. Possible explanations of the poor showing of the NLQS in the ALP field study in contrast to the good results in the KFG field study could be:

Language Dependence. The English syntax of the NLQS has been written on the model of the syntax for German. For example, morphological rules and the user-independent vocabulary were replaced, and the rules for dependent clause word order were deleted. The interpretation routines are the same as in the

German version with some minor modifications. ALP was the first application of the English system version. Therefore the simplest explanation of the high error rates would be that there was still a need for debugging tests.

Database Dependence. While the database schemata of KFG and ALP as well as those of the stage A studies were of comparable complexity (two to six base relations), the size of the ALP database turned out to cause serious response time problems through inefficient translation of natural language into SQL. This does not affect the general concept of the system but stresses the necessity of query optimization in the natural language system.

User Dependence. Since the KFG study was mainly a one-user study, it could be suspected that the main KFG user was a happy coincidence and that the very long usage period and his involvement in the application design provided him with a deeper understanding of the system. On first sight, the fact that KFG was the only application reaching such a low error rate would seem to confirm this assumption. Even the other two KFG users had somewhat higher error rates. However, one has to be cautious: Krause [1982] shows clearly that the main KFG user had few changes in error rates over time, thus denying a learning effect after the initial phase.

Experimental Design Dependence. The application-specific part of the ALP grammar was hardly changed after initial testing, whereas the KFG application was adapted whenever problems became visible in a user session. On one hand, the KFG experience shows that the NLQS is powerful enough to cover the language subset required for a particular application in an impressive manner (93% success). Moreover, it is perfectly acceptable to expect a certain period of time, during which the system has to be adapted to a user. On the other hand, the question arises: when will this user adaptation terminate? The answer is clearly important for the commercial (rather than technical) feasibility of NLI.

Technical Environment Dependence. A final reason for the high NLQS failure rates in ALP is obvious when looking at the EDP protocols: the poor system performance at New York University (caused by slow and noisy communication lines, and system overload), and difficulties with the operating system.

5.0 CONCLUSIONS

The comparison of several experiments with the same domain-independent natural language query system has yielded methodological results and preliminary conclusions about this type of natural language interface, as well as gaps in the studies and opportunities for future research.

Research Methodology. There seems to be a natural sequence to be followed in the evaluation of a natural language query system in order to yield meaningful results. Starting with exploratory on-site system tests, the strategy proceeds towards a qualitative feature analysis, upon which structured quantitative evaluation models can be based. The ALP experience has demonstrated that such a schema can be exploited to its fullest only if the prototype under study has reached sufficient maturity; otherwise, quantitative analyses must be complemented by qualitative studies in order to separate generalizable results from those influenced by the prototype status of the system. It is also critical to provide an adequate technical environment.

Domain-Independent Natural Language Query Systems. Concerning the three introductory questions set forth about domain-independent natural language query systems, some conclusions can be drawn, whereas other issues require further study. Addressing first the desirability question, we know that natural language allows for more concise query input and requires less formulation time than a formal query language. However, nobody has been able so far to demonstrate advantages of natural language over formal query languages in terms of learnability, language power, task performance, or query acceptance rates.

Concerning NLQS feasibility, there is no evidence that any of the experiments exceeded the boundaries of what can be easily implemented within the domain-independent subset system approach. Thus, practice-oriented natural language query systems appear to be technically feasible and able to fulfill the purpose they were developed for. However, additional studies will be required to confirm this result.

The third question asked for the cost of adapting a NLQS to a new application. It is not clear how long the adaptation to an application or a new user will take, or to what degree end users will be able to take over this job from specialists in computational linguistics. The experience with ALP indicates that building and stabilizing a new application needs major linguistic information science (computational linguistic) support. That is, different personnel requirements from those for introducing an end user system based on formal query languages may arise [VASS85].

There are indications that in addition to the performance problems some gaps and inadequacies in the application-dependent part of the NLQS are partially responsible for the high error rates in ALP. There are no hints so far that the general philosophy of domain-independent NLQS is insufficient. But these statements are subject to change pending further evidence.

Acknowledgments: This work is based on several studies in cooperation with the IBM Corporation. The projects would not have been possible without the continued support by members of IBM Heidelberg Scientific Center, in particular, A. Blaser, H. Lehmann, N. Ott, and M. Zoeppritz.

REFERENCES

[BATE83]

Bates, M., Bobrow, R.J., "A transportable natural language interface for information retrieval", Proceedings 6th ACM-SIGIR Conference, Washington, D.C., June 1983.

[CHAP73]

Chapanis, A., "The communication of factual information through various channels", Information Storage and Retrieval 9 (1973), 215-231.

[DAME79]

Damerau, F.J.: "The Transformational Question Answering (TQA) System. Operational Statistics", American Journal of Computational Linguistics 7, 1 (1979), 30-42.

[HARR77]

Harris, L.R., "User oriented data base query with the ROBOT natural language system", Proceedings 3rd VLDB Conference, Tokyo 1977, 303-311.

- [JARK82]
Jarke, M., Vassiliou, Y., "Choosing a database query language", New York University Working Paper Series CRIS #68, GBA 84-39 (CR), submitted for publication, November 1982.
- [JARK85a]
Jarke, M., Turner, J.A., Stohr, E.A., Vassiliou, Y., White, N.H., Michielsen, K., "A field evaluation of natural language for data retrieval", IEEE Transactions on Software Engineering SE-11, 1 (1985a), 97-114.
- [JARK85b]
Jarke, M., Krause, J., Vassiliou, Y., "Studies in the Evaluation of a Domain-Independent Natural Language Query System", in H.Bolc (ed.): Cooperative Interactive Systems, Springer, to appear, 1985b.
- [KRAU80]
Krause, J., "Natural language access to information systems: an evaluation study of its acceptance by end users", Information Systems 4 (1980), 297-318.
- [KRAU82]
Krause, J.: Mensch-Maschine-Kommunikation in natuerlicher Sprache, Niemeyer, Tuebingen 1982.
- [LEHM78]
Lehmann, H., Ott, N., Zoepritz, M., "User experiments with natural language for database access", Proceedings 7th International Conference on Computational Linguistics, Bergen 1978b.
- [MORA81]
Moran, T., "An applied psychology of the user", ACM Computing Surveys 13, 1 (1981), 1-12.
- [REIS81]
Reisner, P., "Human factors studies of database query languages: a survey and assessment", ACM Computing Surveys 13, 1 (1981), 13-32.
- [SCHN84]
Schneider, M., "Ergonomic considerations in the design of control languages", in Y. Vassiliou (ed.): Human Factors and Interactive Computer Systems, Ablex, Norwood, NJ, 1984.
- [SHNE80]
Shneiderman, B., Software Psychology, Winthrop 1980.
- [SMAL77]
Small, D., Weldon, L.J., "The efficiency of retrieving information from computers using natural and structured query languages", Report SAI-78-655-WA, Science Applications, September 1977.
- [TENN79]
Tennant, H.R.: Evaluation of natural language processors, Ph.D. diss., University of Illinois, Urbana 1979.
- [TURN84]
Turner, J.A., Jarke, M., Stohr, E.A., Vassiliou, Y., White N.H., "Using restricted natural language for data retrieval - a plan for field evaluation", in Y. Vassiliou (ed.): Human Factors and Interactive Computer Systems, Ablex, Norwood, NJ, 1984.
- [VASS83]
Vassiliou, Y., Jarke, M., Stohr, E.A., Turner, J.A., White, N.H.: "Natural language for database queries: a laboratory study", MIS Quarterly 7, 4 (1983), 47-61.
- [VASS85]
Vassiliou, Y., Jarke, M., Stohr, E.A., Turner, J.A., White, N.H., "Requirements for developing Natural Language Query Applications", in Languages for Automation, S-K.Chang, (ed.), Plenum Press, 1985 (to appear).

Modeling Natural Language Data for Automatic Creation of a Database from Free-Text Input¹

Naomi Sager*, Emile C. Chi*, Carol Friedman*, Margaret S. Lyman, MD+

*Linguistic String Project,
Courant Institute of Mathematical Sciences, NYU
+ Department of Pediatrics, NYU Medical Center

ABSTRACT

This paper describes (a) computer programs developed by the Linguistic String Project (LSP) of N.Y.U. that map the free-text of technical documents into a semantic representation of document content; and (b) the further mapping of the processed narrative into a database using network and relational data models. The data used in experiments have been primarily hospital discharge summaries and physician's notes on outpatient encounter forms. When the medical narrative is processed and mapped into a database, applications programs can generate summary tables of symptoms, therapies, diagnoses and other features ranging over several dozen variables, and can compute complex relations, such as possible side effects of drugs.

1. INTRODUCTION

This paper shows how narrative documents in technical areas can be analyzed by computer and mapped into a database for retrieval in terms of the informational categories and relations in the narrative. The algorithms are based on fundamental properties of language as a carrier of information that are particularly marked in scientific and technical writing. In such texts, where the primary purpose is to communicate information, the situation is very different than for the language as a whole. The concrete vocabulary is limited to the terms that for the most part have clear denotations, representing the objects of interest to the field, and the types of statements that are made are those that constitute "say-able" (not necessarily true or clear) utterances in the field. In a medical document, for example, one will find words for diseases, symptoms, medications, etc., not airplane parts; and the statements will be of the types that can be said of these objects, e.g., *patient is taking aspirin for rheumatoid arthritis*, NOT *aspirin is taking rheumatoid arthritis for patient*, NOT *aspirin has swollen joints*, etc.

The restricted use of language in a specialized subject area can be characterized by quasi-grammatical rules. The discourse in that subject area is then called a **sublanguage** and the rules its **sublanguage grammar** [HARR68, KITT82]. Syntactic formulas summarizing the well formed statement types for the sublanguage form the informational skeletal structure of the discipline. Each sublanguage formula is in effect a template, or **information format**, for information of a given type [SAGE78]. Sublanguage text sentences can then be mapped by computer into the information formats of the sublanguage [SAGE81], with the result that the information-formatted sentences constitute a formal semantic representation of the information in the documents.

2. INFORMATION FORMATS

Information-formatting has been implemented for the sublanguage of clinical reporting, i.e., narrative portions of patient records, with data drawn from hospital discharge summaries in diverse disease areas [KORE63] and outpatient encounter forms, including the VIS G visit forms of the

¹ This research was supported in part by National Library of Medicine grant number 1-RO1-LM03933, awarded by the National Institutes of Health, Department of Health and Human Services, and in part by National Science Foundation grant number IST83-14499 from the Division of Information Science and Technology.

American Rheumatism Association Medical Information System (ARAMIS).² The VIS G form contains a checklist covering most of the physical examination (about 2/3 page) and additional space for the physician to enter narrative Progress Notes. The coded material is entered into a database. One goal of the study using ARAMIS data is to computer-analyze the narrative material, so that it can be compared with the database of coded material.

The transcribed narrative portion of a VIS G document is input to the text processing system and automatically assigned a code number as shown in Fig. 1. As an example of the results of text processing, Fig. 2 shows a table of information obtained by processing the narrative progress notes seen in Fig. 1. (The Plan section has been omitted). The successive rows of the table correspond to the individual events or observations in the narrative in the sequence in which they appear in the document. Each column holds a particular kind of information, corresponding for the most part to a single sublanguage word class (e.g., a word in the symptom class maps into the SYMPTOM slot of the PATIENT STATE DATA format). Syntactic relations among the columns are not shown in the table but are known to the program; e.g., under PATIENT-STATE DATA, the first column contains the subject of the observation and the second column the predicate (*ankle has/shows fatigue*, in row 3). Like wise, EVIDENTIAL and TIME modifiers are linked internally to the item they modify (the CHANGE item *increased* is associated with *fatigue* in row 3).

3. NARRATIVE PROCESSING

Input to the LSP narrative processor consists of the text of patient documents as transcribed from dictation or the physicians' handwritten reports. In the development to date, editorial conventions have been kept to a minimum so that the input can be truly natural language. Thus, shortened sentence forms and run-on strings of descriptors are tolerated, but some prepositions are filled in (*burning sensation soles both feet* → *burning sensation on soles of both feet*) and some symbols are translated into words (e.g., † into *increased*).

The processing requires a lexicon that provides for each word its major parts of speech (e.g., noun, adjective), its English subclass memberships (e.g., singular, plural) and its medical subclasses (e.g., bodypart, sign/symptom). Currently, about 100 English subclasses and 50 medical subclasses are in use, both to aid in processing (to resolve structural ambiguity), and to represent the semantic content of the material. A preliminary step prior to document processing is to run the words of the documents past the existing computer lexicon and to provide new entries for the words not found. At this writing the medical lexicon contains about 8,000 words, and an English back up dictionary about 12,000 words.

Document sentences are analyzed in five main processing steps consisting of parsing, selection, regularization, information-formatting, and normalization. Previous papers have described four steps. The first module (parsing) has recently been split into two components for a better integration of the sublanguage constraints with the English processing.

Parsing

The parsing component segments the sentence into its major syntactic units and identifies the grammatical relations within and among these units, e.g., which word is the subject and which the object of a given verb, and which strings are modifiers of others. The parser uses a broad coverage English grammar [SAGE81] which has been modified for the special syntax found commonly in medical documents e.g., sentence fragments and run-on sentences.

² VIS G -- Rheumatoid Arthritis, Version 15-220V G (Rev. 1/80), developed by a Committee of the American Rheumatism Association as part of a Uniform Database for Rheumatic Disease. Completed VIS G forms on 50 patients were provided through the courtesy of Dr. James Fries of Stanford University Medical Center, Palo Alto, CA.

Figure 1
 INPUT TO A TEXT PROCESSING SYSTEM
 Pt. 4, Visit 15, ARAMIS Data

HIP04 15.0.0 HISTORY

HIP04 15.1.1 HI - DEVELOPED BURNING SENSATION ON SOLES OF BOTH FEET SINCE LAST VISIT .

HIP04 15.1.2 HI - INCREASED FATIGUE , ACHING , SWELLING OF ANKLES AND FEET .

HIP04 15.1.3 HI - HB HAS DECREASED FROM 13 TO 11 GM .

PEP04 15.0.0 PHYSICAL EXAM

PEP04 15.1.1 PE - NO EVIDENCE OF VASCULITIS .

PEP04 15.1.2 PE - NO MOTOR WEAKNESS .

PEP04 15.1.3 PE - STOCKING DISTRIBUTION SENSORY LOSS IN BOTH FEET.

IPP04 15.0.0 IMPRESSION

IPP04 15.1.1 IP - RA WITH INCREASING ACTIVITY OF DISEASE

PLP04 15.0.0 PLAN

PLP04 15.1.1 PL - INCREASE PENICILLAMINE TO 750 MG DAILY

PLP04 15.1.2 PL - RTC 1 MONTH .

PA-TIENT NO.	VISIT NO.	PARA-GRAPH	SENTENCE I.D.	FOR-MAT TYPE	PATIENT-STATE DATA		TREATMENT DATA	TEST DATA		EVIDENTIAL	TIME	
					BODYPART/ BODYFUNCTION/ BODY MEASURE	SIGN/ SYMPTOM/ DIAG	MED MGMT	TEST-TYPE	QUANT		EVENT-TIME	TIME-ASPECT CHANGE/BEGIN/END
4	15	HISTORY	HIP04 15.1.1	5	/sensation -sole (on) --foot (of) ---both	/burning					since visit (see next line)	/develop
			HIP04 15.1.1	1			visit				last	
			HIP04 15.1.2	5	ankle (of)	/fatigue						increased
			HIP04 15.1.2	5	foot (of)	/fatigue						increased
			HIP04 15.1.2	5	ankle (of)	/aching						increased
			HIP04 15.1.2	5	foot (of)	/aching						increased
			HIP04 15.1.2	5	ankle (of)	/swelling						increased
			HIP04 15.1.2	5	foot (of)	/swelling						increased
			HIP04 15.1.3	4				Hemo-globin	from 13 to 11 mg			decrease
		PHYSICAL EXAM	PEP04 15.1.1	5		/vasculitis (of)					no evidence	
			PEP04 15.1.2	5	motor	/weakness					no	
			PEP04 15.1.3	5	/sensory -stocking distribution --foot (in) ---both	/loss						
		IMPRES-SION	IPP04 15.1.1	5		//rheumatoid arthritis						
			IPP04 15.1.1	5		//disease-activity						increasing

Figure 2
 COMPUTER-ANALYZED PROGRESS NOTES: Information Table for Pt 4, Visit 15, ARAMIS data

Selection

The selection component [FRIE83, FRIE84] utilizes medical word subclasses as they may occur in particular syntactic relations to rule out unacceptable analyses in the medical sublanguage, and to identify which particular sublanguage statement type is occurring. Consider the sentence *Patient on aspirin with RA*. *Aspirin* is in the subclass H-MED for medications. *RA* is in the subclass H-DIAG for diagnoses or diseases. As the medical sublanguage does not allow the combination H-MED "with" H-DIAG, the selection module rules out *aspirin with RA* (interpreted as *aspirin has RA*) and attaches *with RA* to *patient*, which is the correct combination.

This component also resolves many sublanguage homographs (words with more than one sublanguage usage, such as *discharge from hospital* vs. *discharge from nose*) by checking the occurring word class sequence against a list of wellformed word class patterns. Thus, in the preceding example, in both cases the correct subclass of *discharge* will be selected from the two assigned it in the lexicon because of its occurrence with words of different medical subclasses in the two cases (*hospital* in the class of INSTitutions in the one case, and *nose* in the class of BODYPARTs in the other).

Regularization

A third stage of processing (regularization) is required because natural language provides alternative grammatical forms for the same information: *pain in left leg*; *left leg pain*; *patient complains of painful sensations in left leg*; *pain, left leg*, etc. In this component, English transformations operate on the output of the parsing stage so as to eliminate all but one of a set of informationally equivalent forms. Also conjunctive constructions are expanded (*Knee is red and swollen* → *Knee is red and knee is swollen*) and connectives are given a uniform operator-argument form where the arguments are elementary assertions or fragments. E.g. a sentence containing a subordinate clause is changed to a structure consisting of the subordinate conjunction operating on two elementary assertions. Thus, in the parse tree obtained for *Patient had fever when he was seen in the ER*, the subordinate clause *when he was seen in the ER* appears syntactically as a modifier of *Patient had fever*. The parse tree for this sentence is transformed into a structure in which the operator is *when* and its arguments are the elementary assertions: *patient had fever* and *he was seen in the ER*. (ER = Emergency Room.)

There is also a set of sublanguage regularization transformations that are executed in this component. Some nounphrases in the medical sublanguage contain two medical events, which later the processor should map into two different formats (= 2 rows of the table). The English function word (preposition) is not sufficient to decide the issue; the sublanguage classes of the nouns must be considered. For example, *headache from fever* contains two events while *headache from last Tuesday onwards* contains one event. A sublanguage regularization transformation breaks up phrases whose composition in terms of sublanguage word classes with particular prepositions indicates that two events should be represented, and gives them the same operator-argument form as two full assertions under a subordinate conjunction.

Information-Formatting

The fourth stage in converting the information in a medical sentence from narrative to structured form is to transform the regularized parse tree of every component assertion into a semantic structure (information format) in which words carrying a particular kind of medical information (usually members of the same medical word class) are assigned a unique position.

In clinical narrative, six main statement types have been distinguished (Fig. 3). The mapping of the parse tree of an assertion into the format for one of these types depends on the medical word classes of the words occurring in the assertion and their syntactic relations within it. Figure 4 shows a simplified version of the PATIENT-STATE format. Under each node is the crucial word subclass (or subclasses) that determines which words can occur as values of the node (with examples). The nodes PTSTATE-SUBJ and PT-STATE represent the subject-predicate relationship of the assertion. The four nodes under PTSTATE-SUBJ carry values of the alternative options for the subject; those under PT-STATE carry the five options for the predicate. EXAMTEST and TESTRES nodes are

FIGURE 3

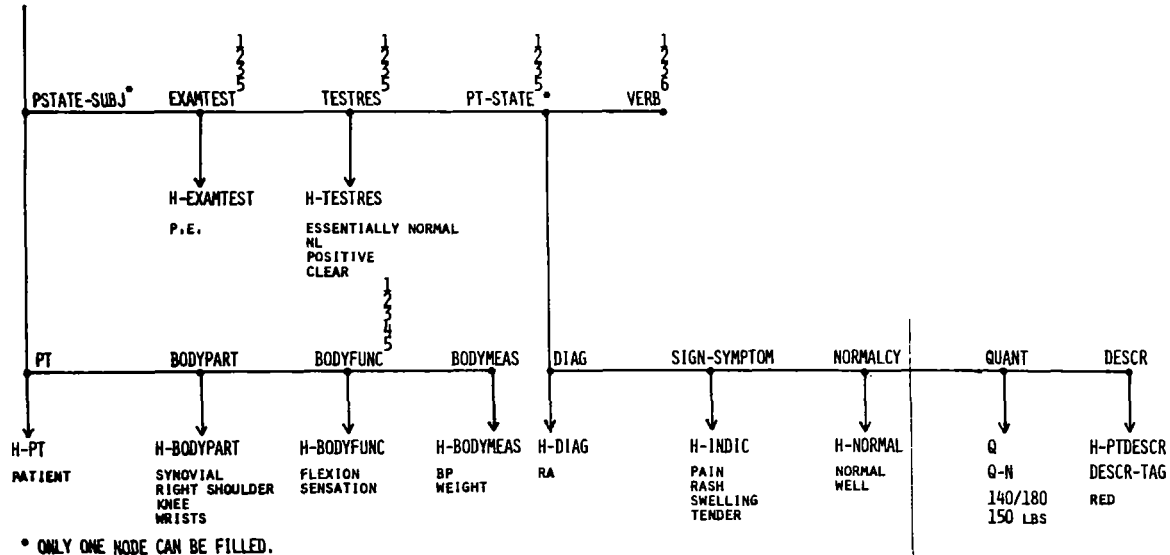
CLINICAL NARRATIVE FORMAT TYPES

FORMAT TYPE	NUMBER	EXAMPLE
GENERAL MED. MGMNT.	1	PATIENT WAS SEEN IN THE EMERGENCY ROOM
TREATMENT OTHER THAN MEDICATION	2	WEARING ORTHOPEDIC SHOES
MEDICATION	3	TAKING ASPIRIN 12 TABS QD
TEST & RESULT	4	RA SERO POSITIVE FOR 15 YEARS
PATIENT STATE	5	PAIN IN KNEES AND ANKLES, RT > LT
PATIENT BEHAVIOR	6	HAS SMOKED FOR 30 YEARS

FIGURE 4: FORMAT 5 (PATIENT STATE)

SIMPLIFIED

FORMATS (PATIENT STATE)



for the statement of a physical examination test procedure and its result. Instances of the PATIENT-STATE format were seen in Fig. 2.

Format modifiers (Fig. 5) carry crucial information, characteristic of natural language and difficult to accommodate in check-list type data collection forms. EVENT-TIME is for explicit time mentions in an assertion. TIME-ASPECT covers BEGinning and END of events as well as CHANGE of state and REPetition. MODS carry NEGation, uncertainty (MODAL) or positive EVIDence. Some modifiers may also have modifiers because this frequently occurs in natural language. For example, in *pain did not start to decrease until she took extra strength tylenol, decrease*, a CHANGE, modifies *pain*, a SIGN-SYMPOM; *start*, a BEG, modifies *decrease*; *not*, a NEG, modifies *start*. Also, the richness of anatomical description is carried in nested bodypart modifiers. This nesting must be accurately represented in the formats in order to be correctly interpreted for retrievals.

Normalization

The final stage of processing, normalization, prepares the structured narrative for mapping into a database or for other applications. Implicit information that can be recovered from context is filled in, with special emphasis on time information. One complex aspect of narrative is the structure imposed by the time sequence of events. This partial ordering of medical events in the narrative can be reconstructed to a large extent from the time expressions in the text [HIRS81].

4. DATABASE DESIGN

Mapping information formats into a database imposes further schematization on the information in the original narrative. The structured narrative must be fitted into a data model: into a schema of record-types with ownership relations and set memberships (network model), or into relational tables (relational model). Our first design was a network (CODASYL) model. The other designs have been relational models, using SYSTEM R, RIM, and our own design, which maps the data into a form suitable for querying directly with application programs, and also for straightforward mapping into a DBMS like INGRES. The guiding principle of all but the RIM design has been to map the full tree structure of the information formats into the database, so as to facilitate the implementation of complex queries.

Early Models

The CODASYL design was implemented using the DMS1100 DBMS on a UNIVAC Series 1100 [SAGE80]. At that time, a large single information format, with fields accomodating all the possible combinations of medical word subclasses was used to structure medical narrative [HIRS82]. The format also specified positions for all of the modifiers discussed above. The database design defined a record type, MEDFACT, into which information formats were mapped, labelled according to the different types of medical information present (this is now achieved by the division into the format types seen above). An EVENT record type was defined to correspond to major linguistic units within the MEDFACT record, mainly subject and object noun phrases, and verbs or predicates. A MEDFACT record owned all of the EVENT records belonging to the same information format.

The network design was very useful for showing us how to approach the problem of imposing a standard DBMS structure on linguistically derived information formats. However, it proved very difficult to query. The complexity of the data which is obtained from narrative input leads to queries with more complex logic than are common with typical tabular data. Using the network query processor, QLP, it was difficult to implement retrievals which test for complex combinations of path occurrences in the network.

The first relational model used an experimental version of IBM's relational DBMS SQL/DS on an IBM 4341 running VM370-CMS. Relations (tables) were defined corresponding to the record types of the network design, including an EVENT attribute for each of the format modifiers. Linguistic connectives between elementary assertions were represented with a CONNECTIVE

FIGURE 5
FORMAT MODIFIERS

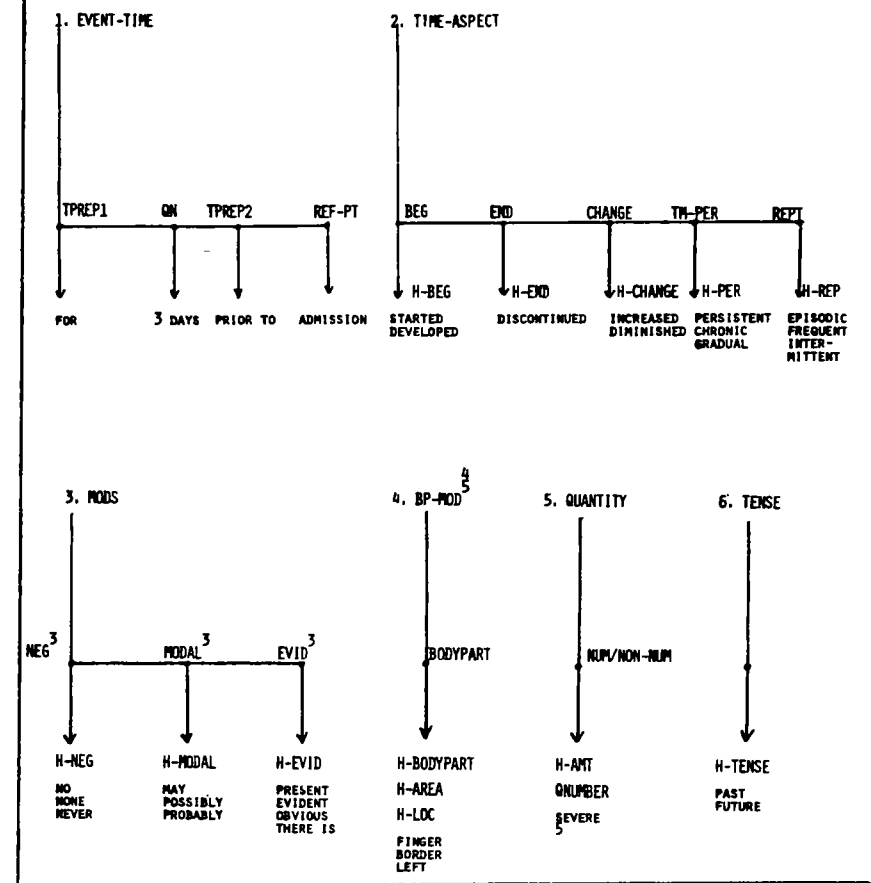
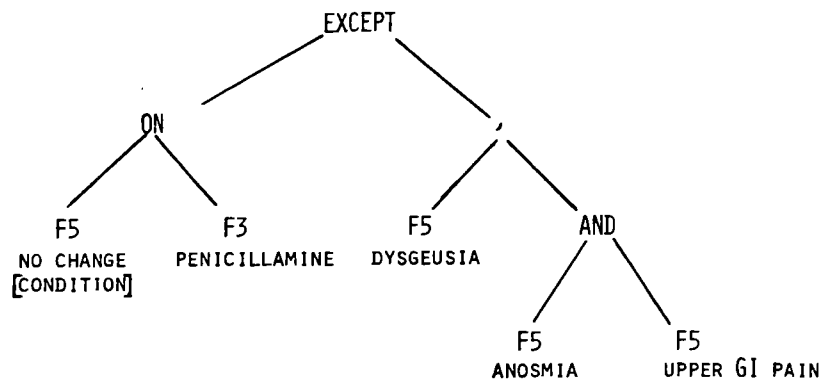


Figure 6

CONNECTIVE TREES

HIP17 5.1.1 NO CHANGE ON PENICILLAMINE EXCEPT DYSGEUSIA, ANOSMIA AND UPPER GI PAIN.



relation, with PARENT, LOFF, and ROFF attributes serving to map the recursive linguistic structure of connectives into a binary tree. (In the network version, connective records were represented outside of the schema due to limitations of the current CODASYL set mechanisms).

Retrieval requests for the SQL/DS database were drawn from a set of 56 test queries of varying complexity culled from health care evaluation criteria and research protocols in the literature. The queries were implemented in SQL, on a small experimental database consisting of hospital discharge summaries [CHIE83]. It proved easy to implement simple queries such as FIND ALL PATIENTS WITH POSSIBLE POSITIVE CHEST X-RAY and possible (although awkward) to implement more complex queries such as: FIND ALL PATIENTS WITH DIAGNOSIS OF PNEUMONIA WHOSE HISTORY OR EXAM AT ADMISSION CONTAINS TWO OR MORE OF THE FOLLOWING:

- (1) CHEST OR ABDOMINAL PAIN
- (2) DYSPNEA OR RESPIRATORY DISTRESS
- (3) RALES IN LUNG(S)
- (4) COUGH
- (5) RIB OR STERNAL RETRACTION

More complex queries involving time relationships in the narrative were also written in SQL. E.g. *Determine whether a bacterial culture is from the first csf (cerebrospinal fluid) sample taken from the patient* [LYMA83]. Time information, represented as a partially ordered graph, is stored by the time program (Module 5 of the processor) in a matrix of time relationships. Its transitive closure, equivalent to a graph of all of the time relationships derivable automatically from the document, is stored as a database table. This table contains the time ordering of any two events mentioned in the text, if the ordering is computable. For example, the above query accesses this table to determine that a mentioned *csf* test does not occur after another *csf* test.

Recent Models

The second relational implementation used Boeing's RIM version 5.0 on a VAX 750 running VMS 4.1. RIM was chosen because it can handle less complex queries readily and is easy to use. In this design, we flattened the format trees to facilitate a simpler mapping into flat relational tables. Whereas this mapping ignored much of the linguistic structure (all connectives and the nesting of modifiers), it facilitated the implementation of simple queries and generation of tables for display. Each node type in each format was used to define a relation in the database. Queries, such as *List all signs or symptoms which show change and possible negation* were readily implemented by intersecting relations.

Our most recent relational implementation uses the full linguistic structure of the format trees. Until a sufficiently powerful relational DBMS was available to us, we mapped the formats into relational tables of our own design [CHI85]. These were designed in such a way as to facilitate further mapping to a DBMS. We coded (in PL/I) a small number of queries to demonstrate the feasibility of using such a relational DBMS. These queries utilize the linguistic relations occurring among individually formatted medical events, and also the discourse relationships between sentences.

One pre-programmed query retrieved all possible drug side effects from the Progress Notes for a selected patient (Table 1). The algorithm for establishing such complex data interconnections in the formatted medical narrative was developed under the close supervision of a medical consultant. The general philosophy of the retrieval was to emphasize recall over precision; i.e., to retrieve all possible side effects for medical personnel to check, rather than risk missing some [CHI85]. It was possible to develop such an algorithm because information formatting preserves the semantic relationships among terms carried in the original text.

Evaluating the format modifiers is critical to the success of these retrievals and a substantial part of the query program is devoted to this evaluation.

For example, to analyze change modifiers, the query program calls on a change evaluation module whenever a CHANGE modifier is encountered. This module attempts to evaluate changes in

Table 1

TABLE OF POSSIBLE SIDE EFFECTS

* Indicates qualified by concessive

DATE	DRUG	EFFECT	RELATION	TEXT ID
02/14/80	NAPROSYN	RASH	ON	HIP17 2. 1. 1
02/14/80	NAPROSYN	RASH		HIP17 2. 1. 1
02/14/80	NAPROSYN	RASH		HIP17 2. 1. 1
02/14/80	PLAQUENIL	SYMPTOMS		HIP17 2. 1. 3
04/10/80	GOLD	STOMATITIS	WITH	HIP17 3. 1. 2
04/10/80	GOLD	RASH		HIP17 3. 1. 2
04/10/80	NAPROXEN	RASH	ON	HIP17 3. 1. 3
04/10/80	NAPROXEN	RASH		HIP17 3. 1. 3
06/11/80	PENICILLAMINE	PAIN	ON	HIP17 5. 1. 1*
06/11/80	PENICILLAMINE	ANOSMIA		HIP17 5. 1. 1*
06/11/80	PENICILLAMINE	DYSGEUSIA		HIP17 5. 1. 1*
06/11/80	PENICILLAMINE	SIDE EFFECT	OF	ASP17 5. 1. 1
08/14/80	ZINC TABLET	DYSGEUSIA	DESPITE	HIP17 7. 1. 2
08/14/80	PENICILLAMINE	DYSGEUSIA	SECONDARY TO	ASP17 7. 1. 1

patient state as GOOD, BAD, or UNCHANGED, (the default is UNDETERMINED), by comparing change direction with the patient state. e.g., *increased pain* is BAD, whereas *increased mobility* is GOOD.

To analyze NEGation modifiers, the program marks the node on which the modifier appears as "negated" (e.g., *no fever* is a negated SIGN/SYMPATOM). MODAL modifiers reflect the fuzziness of the statement of many findings in the sense of [ZADE75] (e.g., *possible drug reaction*). The program marks the node on which such modifiers appear as "fuzzy", though it does not assign a numerical value to the degree of uncertainty suggested by the modal. Modifiers that assert factuality (e.g., *there is, evidence of*) cause the program to mark the associated node as "factual".

Sentences containing a special type of English connective ("concessives") such as *except, although, but* are handled in a special manner. The component assertions under a concessive are not complete without considering the other components. (see Fig. 6 for the structure of HIP17 5.1.1, *no change on penicillamine except dysgeusia, anosmia and upper GI pain*). Formats from such sentences are entered into all tables with a "*" flagging the sentence code, an indication to the user to look at the sentence as a whole. E.g. in the above example, the format for *no change on penicillamine* is entered into the database flagged with "*", as are the formats for *dysgeusia, anosmia, and upper GI pain*. The concessive flags are carried into accompanying retrievals. Thus, *dysgeusia, anosmia, and upper GI pain* are entered into the Possible Side Effects Table (Table 1) as possible side effects of *penicillamine*, flagged with "*" to ensure that the context will be examined by the user.

To check the narrative for possible side effect indications which cross sentence boundaries, the query program will, given a sentence that mentions administration of a medication and contains no patient state information, check the next sentence for sign-symptom or diagnosis information that is not coupled to another medication. We do not look down more than two sentences, because the probability of a correct retrieval becomes too low in this case.

There are a number of cases concerning the type of linguistic connective and the distance between sentences over which a relation can be reliably computed, which are still under investigation.

5. CONCLUSION

Although the technique of information formatting is still very new, we believe that in its application to clinical narrative it offers one solution to the "primary dilemma in the use of a computer-based medical record system..." namely, "how to reconcile the physician's custom of recording free-form narrative on a blank page with the computer's need for structure and a pre-defined vocabulary" [BARN84]. In a busy ambulatory clinic using narrative records, tasks that can not now be done

without enormous expenditure of time and effort could be facilitated by having available a database derived from the narrative records. Such tasks include extracting and summarizing sign/symptom information, or drug dosage and response data; identifying possible side effects of medications; and highlighting or flagging data items for attention in subsequent visits. In chronic disease, such as rheumatoid arthritis, the data accumulated for a single patient may stretch over decades, resulting in a voluminous patient record. The combination of narrative processing (information formatting) with database management techniques could provide a kind of "index" to the contents of a patient's medical record and permit selective review of features of the disease process and its treatment over long periods of time.

While it is too soon to speak of other areas of application besides the clinical sublanguage, it should be noted that the modular design of the language processor encapsulates the areas which have to be changed for a new sublanguage and retains in tact both the English processing and the mechanisms for applying sublanguage constraints. In one instance the system has been found to be portable to another domain [MARS85].

Work of the kind reported here must still be considered experimental, if not pure research. The language processor must be made more "robust" and the entire area of providing a suitable front end for the database of processed narrative is yet to be carefully considered. It is a hopeful sign to us, in view of the magnitude of the enterprise we are embarked on, that these are now the problems before us.

6. REFERENCES

[BARN84]

Barnett, G.O. (1984). The Application of Computer-Based Medical-Record Systems in Ambulatory Practice. *New England J. of Medicine* 310:25, (1984), 1643-1650.

[CHI83]

Chi, E.C., Sager, N., Tick, L.J., and Lyman, M. (1983). Relational Database Modeling of Free-Text Medical Narrative. *Medical Informatics* 8:3 (1983), 209-223. (Special Issue: New Methods for the Analysis of Clinical Data.) Taylor & Francis Ltd., London.

[CHI85]

Chi, E.C., Lyman, M., MD, Sager, N., Friedman, C., and Macleod, C. (1985). A Database of Computer-Structured Narrative: Methods of Computing Complex Relations. *Proceedings of the Ninth Annual Symposium on Computer Applications in Medical Care (SCAMC9)*, Baltimore, MD, November 1985, in press.

[FRIE83]

Friedman, C., Sager, N., Chi, E.C., Marsh, E., Christenson, C., and Lyman, M.S., MD (1983). Computer Structuring of Free-Text Patient Data. *Proceedings of the 7th Annual Symposium on Computer Applications in Medical Care (SCAMC7)* (R. Dayhoff, ed.), 692-695. IEEE Computer Society, Silver Spring, MD.

[FRIE84]

Friedman, C. (1984). Sublanguage Text Processing -- Application to Medical Narrative. In *Sublanguage: Description and Processing* (R. Grishman and R. Kittredge, eds.). Lawrence Erlbaum, in press.

[HARR68]

Harris, Z.S. (1968). *Mathematical Structures of Language*, Wiley-Interscience, New York.

[HIRS81]

Hirschman, L. (1981). Retrieving Time Information from Natural Language Texts. *Information Retrieval Research* (R.N. Oddy, S.E. Robertson, C.J. Van Rijsbergen and P. Williams, eds.), Butterworths, London, 154-171.

- [HIRS82] Hirschman, L., and Sager, N. (1982). Automatic Information Formatting of a Medical Sublanguage. *Sublanguage: Studies of Language in Restricted Semantic Domains* (R. Kittredge and J. Lehrberger, eds.). Series on Foundation of Communication (R. Posner, ed.), Walter de Gruyter, Berlin, pp. 27-80.
- [KIT82] Kittredge, R., and Lehrberger, J. (1982). *Sublanguage: Studies of Language in Restricted Semantic Domains*, Walter de Gruyter, Berlin.
- [KORE63] Korein, J., Woodbury, M.A., Tick, J.L., Cady, L.D., Goodgold, A.L., and Randt, C.T. (1963). Computer Processing of Medical Data by Variable-Field-Length Format. *JAMA* 186 (1963), 132-138.
- [LYMA83] Lyman, M., Chi, E.C., Sager, N., Tick, L.J., and Story, G.A. (1983). Automated Case Review of Acute Bacterial Meningitis of Childhood. In *Proceedings of MEDINFO 83*, Amsterdam, August 1983.
- [MARS85] Marsh, E., and Friedman, C. (1985). Transporting the Linguistic String Project System from a Medical to a Navy Domain. Assoc. for Computing Machinery, in press.
- [SAGE78] Sager, N. (1978). Natural Language Information Formatting: The Automatic Conversion of Texts to a Structured Data Base. In *Advances in Computers* 17 (M.C. Yovits, ed.), 89-162. Academic Press, NY.
- [SAGE80] Sager, N., Tick, L., Story, G., and Hirschman, L. (1980). A CODASYL-type Schema for Natural Language Medical Records. *Proceedings of the Fourth Annual Symposium on Computer Applications in Medical Care 2* (J.T. O'Neill, ed.), 1027-1033. IEEE Computer Society, Los Angeles, CA.
- [SAGE81] Sager, N. (1981). *Natural Language Information Processing: A Computer Grammar of English and Its Applications*. Addison-Wesley, Reading, Mass.
- [ZADE75] Zadeh, L.A. (1975). The Concept of a Linguistic Variable and Its Application to Appropriate Reasoning -- 1, *Information Sciences* 8 (1975), 199-249.

Alternatives to the Use of Natural Language in Interfacing to Databases

Zenon W. Pylyshyn
Centre For Cognitive Science
University of Western Ontario
London, Ontario, Canada N6A 5C2

Despite its attractions, the use of Natural Language (NL) as input to database systems presents a number of problems -- especially in the present state of technology, and possibly in general. Several of these are discussed and some alternative types of interface are sketched -- including the use of expert-systems to train users on both the form and contents of a database, to help them formulate queries, and to help organize the retrieval process. The use of "unintelligent" interfaces, such as menu systems and other hybrid alternatives, is also briefly discussed.

1. Introduction

Natural language has generally been viewed as potentially one of the most important technological advances from the perspective of enhancing access to databases, especially for such nonspecialist or casual users as office workers, managers and consumers. Nonetheless, natural language interfaces have certain limitations. In the first place there is the problem of overcoming the sorts of technical limitations that have already been alluded to in the introductory survey [PYLY85]. There are also a number of other arguments that have been made against the reliance on natural language as an interface to databases. Some of these will be reviewed below. But a very general problem is that the keyboard itself has turned out to be a more substantial obstacle to the widespread acceptance of natural language input than had generally been anticipated [HART83].

One practical problem is that natural language appears to be most suitable for users falling within a rather narrow range of experience and sophistication. It is not, as some might have expected, useful to someone totally new to the use of the database, since such a person has no idea what to ask for nor what limits there are to the way a question can be put (the query system obviously cannot be treated the way one might approach a human librarian or other "information person"). On the other hand, it is also not suitable for someone who has had considerable experience using the database, since such a person would generally prefer to get the information by a more direct route than by having to type a long natural language question.

There are other problems with natural language systems. Even if many of their technical limitations were overcome in certain narrow domains, and even if the keyboard problem were alleviated (say, by developments in speech recognition), queries and searches for specified information would nevertheless still frequently fail. Surprisingly, experiments in libraries have shown that when one examines the cases in which users believe that a search was successful, one finds that a full 67% of these cases failed to find relevant books [BATE77]. This inability to correctly locate relevant material is due to the user's lack of familiarity with the contents and the organizational structure of the

database (in this case the library catalogue). Since the users don't know how data is classified or otherwise structured, they do not know what parameters to specify -- i.e. what questions to ask. What may be needed in such cases is not a facility for expressing what one wants in one's own terms, but rather some help in learning the structure of the information (as well as, perhaps, more natural ways of structuring the information), and some natural way of exploring the database to get an idea of the kinds of information in it and how it is arranged -- i.e. a facility for easier browsing. Natural language systems are generally not suitable for browsing or for cases where the user is not thoroughly familiar with either the content or the organization of the database in question.

In order to deal with the case of complex databases that are unfamiliar to users, or indeed with the general problem of providing aids for exploring databases, we can go even further in either of the two directions discussed earlier: (a) We can make the interface more intelligent by incorporating knowledge of the database contents in the interface, together with some reasoning capability, so that the interface itself can guide the user the way an intelligent assistant might, or (b) we can go further in the direction of viewing the interface as a simple and convenient tool, with all the control residing in the user. The first alternative takes us towards expert-system advisors, while the second takes us towards menus and their variants.

2. Expert Systems as a Database Interface

Expert systems have received more attention in the public scrutiny of artificial intelligence than any other development in that field. Because so much has been written about expert systems and about the successes they have achieved in a large number of areas, from medical diagnosis and prospecting to design and crisis management, we will not attempt to go over this ground here (see, however, [HAYE83], and [WEIS84]). Our discussion of this topic will be confined to a sketch of some of the ways in which expert systems might play a role in interfacing users with databases. There are a number of different ways in which an expert system might play such a role.

- **Query and Search Optimization.** Frequently a complex computational process must intervene between the formulation of a query (in whatever form) and the retrieval of an answer -- especially when the database is large and homogeneous. Retrieval can involve extremely complex searches of the database, covering a variety of different criteria and requiring time-consuming set-intersection procedures. Quite often, however, various domain-specific heuristics could be used to improve the efficiency of these searches, if such "control information" were somehow brought to bear on the retrieval problem. In such cases, expert systems could be of help, for example, to recast the internal form of the query so as to lead to a more efficient search, or for optimizing the process of answering a sequence of queries by storing intermediate results and making meaningful connections. [JARK83], [KELL77], and [KELL82] provide examples of such uses of expert systems in enhancing retrieval from databases.
- **Inferring facts not explicitly stored.** In this case the expert system might have the requisite knowledge to answer a question which is not

explicitly retrievable from the database by logically inferring that there is a different question that produces the same answer and is retrievable. Take, for example, the query to list all students who have programming skills equivalent to that provided by course CS20, whether or not they have taken that course. Such information would very likely not be explicitly encoded. Yet it is derivable from other facts, such as that certain other courses (e.g. all computer science honours courses) have that very skill as their prerequisite. In general, if the expert system has a collection of general facts about the database in the form "all Xs are Ys" or "no Xs are Ws" then it is in a position to answer database queries that might not be directly retrievable, or perhaps only answerable by exhaustive enumeration of cases.

- **Database Advisor.** A large number of the expert systems that have been designed actually play the role of advisors (say for diagnosis or programming or financial planning). Thus it is natural to expect that they could serve in a similar capacity in the case of database retrieval. In this case it would be playing a role analogous to that played by a librarian in providing advice on how to go about locating some piece of information. A librarian's role is particularly valuable if the user knows very little about the organization of the library or if the information sought is not clearly defined (e.g. "I would like some information that could help me in deciding on a career"). What is required for a database advisor is an expert system that contains "meta-knowledge" or knowledge about what kinds of topics are covered in the database and how they are organized, cross-classified and accessed. It also could involve knowledge of what is called the "data model" underlying the database design, as well as the retrieval procedures embodied in the database system or query language. Such a "meta-database" expert system could help users formulate their query, could advise them on possible ambiguities or unanticipated consequences of some particular way of querying the database, could explain why some particular query produced certain unsatisfactory results and suggest alternative approaches. In addition, it could serve as an instructional tool to tutor the novice user on the nature and structure of the database or on the use of some particular artificial query language for obtaining information. This type of interface can be designed to contain a substantial component that is independent of the particular database in question. Such an interface, which has been referred to as a *User-Agent* in contrast with a device-dependent *Front-End*, (see [HAYE80]) has great potential.

An example of the use of artificial intelligent techniques, along with findings from empirical studies in Cognitive Science, is a system called RABBIT, developed by [TOU82] at the Xerox Palo Alto Research Center. RABBIT is an intelligent database assistant that helps users formulate a query by interacting with them during the construction of a description of the target item(s). The system infers a *perspective* for the query from the (partial) description provided and from an examination of the database itself. A *perspective* is a way of viewing objects that highlights certain of their attributes. From this the system retrieves a best-match instance and displays its perspective-relevant attributes. The user can then criticize the instance and use its

description as a guide in reformulating the next iteration of the query. For example, the user can point to a descriptor in the query or in the instance that was retrieved and specify that certain of its attributes be treated as *necessary* or as *prohibited*. In addition, the user can ask for *alternatives* or *specializations* of a given descriptor to be used instead. The RABBIT system makes use of a variety of artificial intelligence techniques (e.g. It uses a knowledge representation system called KL-ONE), together with results of psychological studies of human memory (it is motivated by a theoretical hypothesis called "retrieval by instantiation" or "retrieval by reformulation", see [WILL81] in order to help resolve many common difficulties faced by users of database query systems.

3. "Unintelligent" Interfaces: The Computer as a Tool

Earlier we suggested that there are two basic approaches in making large unfamiliar databases more accessible to users. Above we described an approach based on providing the user with an interface that knows a lot about the database -- an "intelligent" interface. If we go to the other extreme, that of providing the user with an interface that has a minimum of autonomous knowledge and inference ability, there are a number of different alternative strategies that might be employed. In all cases our ultimate goal is the same as the one we had when we considered building expert systems; to provide a facility for those users who are not very familiar with the database in question. This time, however, we do not endow the interface with the ability to reason and to perform as a flexible active intelligent assistant. Instead, all the intelligence goes into the initial design: It is "frozen" into the structure of the tool, which remains under control of the user. There are several classes of approach one can take, each of which can help the user browse through the database.

1. The content and structure of the information in the database can be made more visible to the user in the course of interactions with the database. One obvious way of doing this is by providing the user with *part* of the information and relying on *recognition* ability to select a search path, as opposed to requiring the user to generate the entire query expression. The paradigm example of the use of this technique is, of course, the use of *menus*. Below, we will sketch some of the problems associated with the use of menus, and we will describe a major effort at building menu-based systems that has achieved considerable success with large practical databases.
2. The range of queries that can be addressed to a database on each query cycle can be explicitly limited in a natural way. It would not do, of course, to have a query system which *looks* as though it can deal with a wide range of queries but merely rejects most of them. Indeed this was traditionally one of the fatal shortcomings of quasi-natural language query systems. It is far better to have a system which is intrinsically limited by design in such a way that the user knows that only certain clearly delimited queries can be formulated. Menu systems have this quality, as do various key word systems and their extensions in simple artificial query languages. We will discuss these alternatives briefly later.

3. Finally, it is sometimes possible to greatly enhance the access to databases without using exotic computational techniques if one understands better the nature and source of limitation in existing key word and menu systems and the types of information and data structures for which they are most suitable. There is even some reason to believe that varying a few design parameters can dramatically affect the usefulness of these systems -- parameters such as the response time, the frame or menu contents, the choice of key words, or the mechanics of making selections (e.g. by allowing selections to be made by using a pointing device instead of requiring typing). Furthermore, it is sometimes possible to combine the virtues of one or more constrained systems to produce a design that is very effective in certain contexts. We shall examine several of these ideas below.

4. Menu Systems

4.1. Natural Language Menu Systems

Before considering the case for menu systems in general, we examine one special case, since in certain circumstances it provides some of the advantages of a natural language query system. That is the natural language menu-driven system. Unlike the natural language query system described earlier, where the user interacts with the language system in sentential units, the menu driven system incrementally generate parts of the query sentence, and provides a set of phrasal continuations at several stages of the sentence generation. At each stage the user simply selects, from options presented on a partitioned screen, an appropriate phrasal continuation in an incremental construction of a sentence expressing his desired query. The system increments the partial parse and again presents a set of (usually different) continuations appropriate to the new parse in the context of the structure and content of the database. The process continues until the user's query is complete, at which point a response to the query is generated. Because this work [TENN83] is described elsewhere in this volume it will not be described in detail here. Suffice it to point out that as long as the particular phrasal continuations provided by the system are well matched to the user's conceptualization of the intended query, a menu driven system has some obvious advantages over a complete natural language system.

There is, first, the obvious advantage that it is computationally a dramatically simpler system. Users can build them for their own applications in a matter of hours rather than months. The computational resources required to run the system are orders of magnitude less and most systems implemented have been for modest personal computers. Because of their simplicity they tend to be robust and free of bugs. Several different menu systems can be built for the same database, thereby allowing customization for individual needs, as well as controlling access privileges. Menus can be treated as data objects in their own right and can be "granted" to different classes of users.

Although the problem of "coverage" is still a real problem for the system design, as it is for the natural systems discussed earlier, the problem of density of coverage is not. With menu driven incremental sentence generation it is trivially impossible to

generate a non-sentence of the sub-language. The issue of ambiguity is less clear. Ambiguity can certainly be minimized by careful crafting of permissible phrasal continuations. On the other hand there is always a great deal of syntactic ambiguity in natural language systems which native speakers are not aware of because they unconsciously filter out the unintended readings using knowledge of the world. The extent to which problem can be dealt with bringing in semantics or without elaborating the phrasal continuation menus to the point where they become unacceptable large, is very much a function of the sub-language domain. As the complexity of the application increases there comes a point at which the number of continuations, and hence the complexity of the menu, becomes self defeating. Furthermore, it appears that there is evidence that limited systems such as ones based on menu driven natural language queries do have a variety of other limitations [HEND83].

4.2. Practical Large Scale Menu Systems

Menu systems have two disadvantages as normally implemented. First, they are slow. In menu selection, according to some rough estimates provided by [ROBE81], a user can select about one out of ten alternatives every five seconds (i.e. a channel capacity of about 1 bit per second). On the other hand a skilled typist can type about one word per second (i.e. a channel capacity of about 10 bits per second) -- about an order of magnitude faster. The second disadvantage is the forced interposition of explanatory text and options which further compounds the speed problem. Other disadvantages arise when the material does not lend itself well to a hierarchical taxonomy. In that case the user may take a wrong path and miss the target entirely. If the material is not naturally hierarchical, on the other hand, the user can become hopelessly lost in a maze of connections -- a phenomenon also shared by other sequential search methods.

Most of the menu systems one sees are for rather small systems. It is of some interest, therefore, to examine briefly the experience of designing and using a large practical menu-based access system in which some effort was made to overcome at least the obvious difficulty with menu systems and to extend them with some additional features. One of the most sophisticated of the large practical scale menu systems is called ZOG. It was developed at Carnegie-Mellon University as an experiment in user interfaces, and has been used for a wide range of large-scale practical applications [ROBE81].

ZOG is viewed by its designers as a clear case of the evolution of user interfaces in the direction of *tools*, as opposed to the intelligent agents discussed in the preceding section. It is designed to overcome the two main difficulties with menu systems identified above (viz., speed, and the forced interposition of a large amount of text and options). The idea is to overcome these limitations by (a) designing the system to have very fast response time (in the order of 0.5 seconds, or even less, to restore part of the speed imbalance between selecting and typing), and (b) to have a very large well-designed network of small frames which are tailored to permit alternative "short cut" sequences for more experienced users. The design principles behind ZOG are that it should have a rapid response, that making selections should be physically simple, that the database should consist of a large practical-sized network that is hierarchically organized, that individual frames should be simple and easy to modify, that the system

should be transparent and usable with a wide variety of databases and programs, and that it should allow active operations (such as running programs or text editing) as well as operations for moving through a tree.

Evidence of the viability of the design is the practical success of the system in a wide range of large applications. These include a library browsing system; document preparation system; an information system containing facts about the Carnegie-Mellon computer science department, its academic regulations and available software; a computer program development system; a project management system; a data management system for a large expert system application, and a shipboard operations management system for a nuclear powered aircraft carrier (USS Carl Vinson).

4.3. Other considerations in constrained interfaces

The one alternative retrieval scheme in relatively wide use that we have not discussed is the use of key words. The use of key words represents a constrained retrieval system, in the sense discussed earlier. Although the number of words in a natural language may be large (upwards of 50,000) it is still tractable. Indeed it is easy to construct a table lookup algorithm for this number of words even on a microcomputer. What is much more difficult is arranging for a query system to respond to this number of subtle distinctions in a meaningful way. Simply retrieving records or pages that contain the key word is clearly unsatisfactory since the inventory of words contained in such records is usually a very poor indicator of the content of these records. Words are highly ambiguous in their meaning and the same meaning can in general be conveyed in myriad ways using quite different words. There are cases in which key words can be relatively effective, however, and the recognition of such cases can be improved by research on people's choice of descriptors for different subject areas (see, for example, the work of [LAND83]).

As we mentioned above, one of the advantages of menu systems is that over a period of time they allow the user to develop an idea of the structure and content of the database. Thus it would be useful if at that point they provided a vehicle for a transition to other forms of retrieval in order to short-circuit some of the long paths normally necessary in menu explorations. The ZOG system does provide a number of ways of getting around frames without going through the options provided (these involve use of the global selection pads, such as a global search facility). Another alternative might be to allow the use of key words at certain points in the menu-based exploration, with provision for returning to the point of departure from the menu. The only documented case of using such a scheme is one investigated by [CHAO82], who reports that a hybrid scheme is superior to a strict tree structure, with subjects accessing fewer pages and committing fewer errors. As Chao herself points out, however, it is not clear what would happen if either (or both) the size of the database or of the index list were to be expanded significantly, say to the size of the databases used with the ZOG system.

Constrained interfaces of different sorts have been used ever since databases were first invented. In fact the most common types of access schemes for databases continue to be such things as menus, key words, and various formal query languages. Each of these has some advantages and some drawbacks. In some cases, and for certain types of applications, these limitations may not be serious. In other cases a combination of

limited methods, or purely mechanical improvements in the layout of the screen or the speed of response, or simply a graduated tutorial introduction to the use of the interface, can more than make up for the minor inconveniences involved. There is even evidence that explicitly *hiding* certain features from users at first (the obvious metaphor here is the use of "training wheels" to teach bicycling skill) improves their ability to learn to use a computer tool -- presumably by not discouraging or frightening them at the start. Thus there is room for imaginative exploration of interface design, even where the designs are not very general or the interfaces very "intelligent" in the sense of incorporating knowledge of the user. Clearly, what is needed is a systematic investigation of the effects of various aspects of these limited systems in different settings.

REFERENCES

- [BATE77] Bates, M.J. "Factors Affecting Subject Catalog Search Success", *Journal of the American Society for Information Science*, 161-169, May, 1977.
- [CHAO82] Chao, G. "The Use of Keywords in Videotex Systems", for the Department of Communications, Ottawa, Canada, Contract Serial No. OER82-05055 and OER81-05064, 1982.
- [HART83] Hart, P.E., "Directions for AI in the Eighties", *SIGART Newsletter - ACM*, 79: 11-16, January, 1983. [HAYE80]
Hayes, P., Ball, E. and Reddy, R. "Computers with Natural Communication Skills," *Computer Science Research Review: 1979-80*, Carnegie-Mellon University, 1980.
- [HAYE83] Hayes-Roth, R., Waterman, D., and Lenat, D. *Building Expert Systems*, Addison-Wesley, 1983.
- [HEND83] Hendler, J.A., and Michaelis, P.R. "The Effects of Limited Grammar on Interactive Natural Language", *CHI '83 Proceedings*, 190-192, December, 1983.
- [JARK83] Jarke, M. and Vassiliou, Y. "Coupling Expert Systems with Database Management Systems", *Proceeding of the NYU Symposium on Artificial Intelligence Applications to Business*, New York, 1983.
- [KELL82] Kellogg, C. "Knowledge Management: A Practical Amalgam of Knowledge and Data Base Technology", *Proc. AAAAI-82*, 1982.
- [KELL77] Kellogg, C., Klahr, P., and Travis, L. "Deductive Methods for very Large Databases", *Proc. Fifth IJCAI*, M.I.T., 1977.
- [LAND83] Landauer, T.K., Galotti, K.M., and Hartwell, S.H. "Natural Command Names and Initial Learning: A Study of Text Editing Terms", *Comm. ACM*, 26:7, 1983.
- [PYLY85] Pylyshyn, Z. W., and Kittredge, R. I. Databases and Natural Language Processing. *Database Engineering*, 1985 (this issue).
- [ROBE81] Robertson, G., McCracken, D., and Newell, A. "The ZOG Approach to Man-Machine Communication", *International Journal of Man-Machine Studies*, 14, 461-488, 1981.
- [TENN83] Tennant, H.R., Ross K.M. and Thompson, C.W. "Usable Natural Language Interfaces Through Menu Based Natural Language Understanding", *CHI'83 Proceedings*, 154-160, December, 1983.
- [TOU82] Tou, F.N., Williams, M.D., Fikes, R.E., Henderson, D.A., and Malone, T.W. "Rabbit: An Intelligent Database Assistant", *Proceedings of the National Conference on Artificial Intelligence*, 314-318. Menlo Park, CA: AAAI, 1982.
- [WEIS84] Weiss, S.M., and Kulikowski, C.A. *A Practical Guide to Designing Expert Systems*, Totowa, N.J.: Rowman and Allanheld, 1984.

MENU-BASED NATURAL LANGUAGE INTERFACES TO DATABASES

Craig W. Thompson

Texas Instruments, Inc.
PO Box 226015, MS 238
Dallas, Texas 75265

ABSTRACT. "Menu-based natural language", as implemented in the NLMenu system, provides useful near-term solutions to a number of problems that affect conventional natural language interfaces to databases. This article overviews our research on menu-based natural language, describing 1) the basic NLMenu approach, 2) advantages of the approach including ease-of-use for end-users and low cost for interface designers, and 3) applications of the approach for database updates, requests for business graphs and map displays, and mixed dbms and keyword-based information retrieval queries.

1. Menu-Based Natural Language


In a "menu-based natural language" interface [TENN83a, TENN83b], a user is presented with an arrangement of menu panes containing lexical items. Figure 1 shows a sample NLMenu interface to a database of Austin restaurants where the user has phrased the query:

EX Find restaurants whose distance from UT in miles is less than 0.5

The user constructs a sentence (query or command) by selecting a sequence of words or phrases from active windows (those shown with white backgrounds). A semantically constrained grammar provides the look-ahead control structure that activates and highlights those menus containing legal completions of the sentence. This control structure enforces that only understandable sentences can be formed. An implemented system called the NLMenu system has been designed to explore this approach. NLMenu was first developed on Lisp Machines (LMI CADR and LAMBDA, Symbolics 3600s, and TI Explorers) and then ported to TI PCs, where the system is called NaturalLink and is implemented in C.

For ease-of-use, NLMenu interfaces have advantages over both conventional formal database query languages and also conventional natural language interfaces. Conventional query languages require that a user understand and remember the query language syntax and semantics and also the names and relationships of the domain entities and attributes. In NLMenu, this sort of information is available directly in the menus, and natural language provides an immediately understandable control structure for asking even fairly complicated queries. The advantage of NLMenu over conventional natural language interfaces is that the user is guided to use just the subset of natural language that the system "understands". Users of conventional natural language systems often have trouble phrasing their queries, either overshooting or undershooting the capabilities of the interface [TENN80]. Trying to stay within the covered lexicon, syntax and semantics can become a frustrating end in itself. Because of the guided approach employed, NLMenu grammars do not have to cover all paraphrases of a sentence and so can be

Figure 1: An NLMENU Interface to a Restaurant Database

NLMENU Interface Austin Restaurants			
Commands	Nouns	Experts	Modifiers
Find Draw Delete Insert	restaurants (& new restaurant) a bar chart a line graph  pie chart a histogram a scatter plot a surface graph	<specific map locations> <specific restaurant names> <specific addresses> <specific telephone numbers> <specific kinds of food> <specific reviews> <specific qualities of foods> <specific prices> <specific credit cardss> <specific number>	whose map location is whose name is whose address is whose telephone is whose kind of food is whose review is whose quality of food is whose price is whose credit cards are whose distance from ut in miles is with a minimum slice size of with grid on with horizontal grid with vertical grid with <n> divisions
Attributes distance from ut in name address telephone kind of food review quality of food price credit cards	Comparisons between greater than less than greater than or equal to less than or equal to equal to	Connectors and draw a map of them and or	
System Commands			
Restart	Refresh	Rubout	Exit System
Save Input	Retrieve Input	Delete Inputs	Play Input
Show Input	Show Parse Tree	Execute	Save Output
Find restaurants whose distance from ut in miles is less than 0.5			
<i>More Above</i>			
NAME: San Miguel LOCATION: 2830 W. North Loop TELEPHONE: 459-4121 DISTANCE_FROM_UT: 0. KIND-OF-FOOD: MEXICAN REVIEW: ...many favorites on the menu...friendly service. -- Texas Monthly X-LOC: 2500. Y-LOC: 2500. QUALITY-OF-FOOD: GOOD PRICE: MODERATE CREDIT-CARDS: DC, MC, V ICON: ?			
NAME: Connan's LOCATION: Several locations TELEPHONE: 476-1981 DISTANCE_FROM_UT: 0. KIND-OF-FOOD: PIZZA REVIEW: Chicago style deep dish pizza...are impressive. -- Jensen's Austin Guide X-LOC: 2500. Y-LOC: 2500. QUALITY-OF-FOOD: GOOD PRICE: MODERATE CREDIT-CARDS: AE, MC, V ICON: ?			
NAME: Foothills of Austin LOCATION: Hyatt Regency Hotel TELEPHONE: 477-1234 DISTANCE_FROM_UT: 0. KIND-OF-FOOD: CONTINENTAL REVIEW: ...good place for continental cuisine. -- Jensen's Austin Guide X-LOC: 2500. Y-LOC: 2500. QUALITY-OF-FOOD: GOOD PRICE: EXPENSIVE CREDIT-CARDS: AE, MC, V ICON: ?			
3. tuples retrieved			
Nlmenu Display Window Austin Restaurants			
<i>More Below</i>			

considerably smaller, small enough to fit comfortably on a personal computer. Other advantages of NLMenu include no spelling or syntax errors and building a sentence by "recognizing" it instead of "creating" it. Finally, the NLMenu guided approach provides a straightforward way of managing the changing part of the lexicon that corresponds to stored database values. Interaction experts [THOM84c], like the "<specific restaurant names>" expert in the "Experts" window of Figure 1, provide field-specific pop-up menus where help in specifying database values can be located. Neither conventional natural language interfaces nor conventional query languages provide the user with support for specifying valid database values during query or command composition.

2. Building NLMenu Interfaces

A variety of menu-based interfaces have been built manually, interfacing to such diverse target systems as the Dow Jones News and Retrieval Service, the MS/DOS operating system on the TI PC, a guided version of SQL, and a simple expert system. In addition, a grammar writer's workbench has been constructed which aids an interface designer in building interfaces: finding dangling references in grammars and undefined lexical entries, checking rule syntax, generating sample strings from the grammar, and so on [THOM85a].

In the case of interfaces to relational databases, we have constructed an interface generator that can automatically generate usable natural language interfaces for querying a collection of tables [THOM83].

The interface generator takes two inputs. The first is a domain-dependent specification called a "domain spec" which lists, for a given interface, data dictionary information including the tables to be covered, the access rights for each table, a categorization of the attributes of tables in terms of non-numeric, numeric and coded fields, table keys, and a specification of supported join paths. End users supply this argument either by providing a source file spec or by using a menu-based interaction with the target system's data dictionary. The second argument is a domain-independent generic grammar and lexicon targeted on a particular database query language. End-users do not generally modify this argument. The generic grammar and lexicon consist of rule templates, and the domain spec is used to instantiate the templates to generate context free grammar rules and actual lexical entries. For instance, a grammar rule template like:

```
<rel>-mod --> whose-<rel>-<attr>-is <rel>-<attr>-expert
      where (<rel> <attr>) is an element of non-numeric-attributes
```

is instantiated with the domain spec category non-numeric-attributes:

```
non-numeric-attributes =
((restaurant name)(restaurant address)(restaurant telephone)
 (restaurant kind_of_food)(restaurant review)
 (restaurant quality_of_food)(restaurant credit_cards))
```

resulting in seven instantiated semantic grammar rules, providing seven ways to post-modify restaurants, one for each of the seven (<rel> <attr>) pairs in non-numeric-attributes. For instance, when rel = restaurant and attr = name, the following instantiated grammar rule results from the substitution:

restaurant-mod --> whose-restaurant-name-is restaurant-name-expert

Thus a domain-independent generic grammar is merged with a domain spec to produce a semantically constrained grammar.

The separation of the domain-dependent and domain-independent portions of an interface makes it possible for trained but linguistically naive end-users to build their own interfaces. It took only about thirty minutes to build the Austin Restaurant interface. Unlike conventional portable natural language interfaces like Intellect [HARR83], Team [GROS82], and Ask [TB&F83], the generated interfaces are immediately usable. Conventional interfaces require a long empirical tuning phase in which a habitable application-dependent sublanguage is discovered, often requiring that a trained interface designer spend over a man-month of time to build a reasonably useful interface to a specific application. In addition to being portable across applications, the NLMenu interface generator is portable across target database management systems. It takes only a few days for a trained grammar writer to revise a generic grammar so that translations refer to a new target database. Currently implemented target translations include SQL, Ingres Quel, Prolog, and an Explorer Lisp Machine relational table management system called RTMS. The interface generator described above binds the semantics to the generic grammar at interface creation time, building a semantic grammar. Recently, we have developed a dynamic constraint lookahead parser in Prolog which shifts this binding to execution time, making it possible to change the interface description on the fly.

3. Extending the Applicability of Natural Language Interfaces

Initially, the NLMenu interface generator was limited to allowing users to only retrieve data into tables. One of our development goals has been to extend the interface generator so that, in a domain independent manner, extended sorts of functionality can be requested using a broadened menu-based natural language sublanguage.

The first area we examined involved using menu-based natural language to specify database updates. For several reasons, few natural language interface efforts have permitted users to request database updates. First, updates, unlike retrievals, may affect the state of the database in ways that the user did not intend and may be hard to retract. Second, underlying database management systems have differing update policies with respect to updating views, tolerating null values, and supporting transactions. It may be hard for a natural language interface to hide these differing policies. In [THOM84b], it is argued that the update policies of a target database can be reflected in the NLMenu guided approach to guarantee that:

- o only updatable views and tables can be updated
- o required key information must be specified and non-key information is optional
- o CODASYL-like cascaded deletion semantics can be accommodated and complex objects that span several relations can be inserted according to entity and referential integrity constraints

- o operationally-defined transactions for an application can be specified
- o integrity constraints and protection constraints can be specified.

Conventional natural language interfaces, in contrast, provide no way to limit the user to the idiosyncratic update policies available in a given target database management system.

The second area has involved extending the coverage of NLMenu database interfaces to include some graphics [THOM85b]. First, we added some spatial database queries to NLMenu so that a user could ask:

EX Find restaurants whose location is <specific map locations> and whose type of food is Mexican or Chinese and draw a map of them

During query specification, when the user selects the "<specific map locations>" expert item, a map is displayed, showing icons of restaurants on a street map of Austin, allowing the user to use the mouse to rubber band a selected area. When the query is evaluated, the same map is displayed showing icons of just the restaurants selected in the query, allowing the user to zoom and to mouse restaurants and see their attributes. Some distance queries were added as well including the one shown in Figure 1. Later, we added several sorts of business graphs allowing the user to build queries like:

EX Draw a histogram by distance from UT in miles for restaurants whose credit cards are American Express or MasterCard

To add the graphics queries, new constructions were added to the generic grammar and new categories were added to the domain spec to allow units to be specified for dimensional attributes and to allow map backgrounds and graphics icon generators to be associated with relations. Most recently, we have added geographic information system queries like:

EX Draw a map of Texas showing interstate highways which pass through Dallas

where an operator like GRAPHICS-INTERSECTP is the theta-join predicate between a CITY and a HIGHWAY relation over the city-polyline and highway-polyline attributes, and the join predicate and related phrase ("which pass through") are added to the join-path category of the domain spec.

The third area we have examined has involved augmenting RTMS so that information retrieval-like boolean queries on attributes can be specified as part of a relational query [THOM86]. Extending the NLMenu interface generator to allow the user to ask mixed database and information retrieval queries involved only specifying which fields were to be treated as information retrieval fields so that the user could then ask queries like the following, (where "?" and "(w)" are DIALOG-like pattern designators for wild-card and adjacency):

EX Find courses that are taught by professors whose rank is teaching assistant and whose course description involves information(w)retrieval or database? or data(w)base?

The extensions to the basic NLMenu interface grammars mentioned above were domain-independent. Thus, when new interfaces are created, they immediately inherit the extended capabilities described. The ability to create NLMenu interfaces easily opens a need for managing interfaces. To this end, we have explored operations on interfaces including CREATE, MODIFY, DESTROY, GRANT, REVOKE, and COMBINE. Granting and revoking database interfaces is interesting because the capability reflects the idea that an application view usually consists of a set of entities and capabilities and can be granted all at once instead of piecemeal as in SQL. Combining interfaces that originally targetted on different database management systems is interesting because the resulting NLMenu interface makes the problem of querying the different databases transparent to the user.

4. Interface Design Considerations

An experienced NLMenu interface designer learns that there are some interesting and non-obvious design decisions that affect the usability of NLMenu interface designs [THOM84a]. Only two are mentioned here. The "big menu" problem is inherited from menus and occurs whenever too many menu items populate a menu to allow a user to locate desired items or distinguish between related items. Partial solutions to the problem include breaking menus apart functionally, grouping items within a menu hierarchically, allowing menus to scroll, providing help associated with each menu item, and allowing user typein to search menus. The "weak bridge" problem is inherited from grammars and occurs when a construction is introduced with a word or phrase which does not provide enough context for the user to recognize it as the start of the construction. The solutions here are to re-phrase such constructions where possible or to train the user to recognize the weak bridge. Our experience indicates that, while the "big menu" problem and problems of complicated grammars may eventually limit NLMenu's use for some applications, applications as big as fourteen tables and seventy attributes can be accommodated in a single interface [Ster85] and that bigger interfaces can be partitioned into smaller ones.

5. Conclusions

Menu-based natural language is a useful interface technology with several advantages over conventional natural language interface technology. While we have explored several aspects of the approach, several directions appear before us, among them the tradeoff between cooperative response and query optimization, human factors experiments into ease-of-use and ease-of-interface-creation, and applicability of NLMenu to expert systems.

References

[GROS82] Grosz, Barbara, Doug Appelt, Alex Archbold, Bob Moore, Gary Hendrix, Jerry Hobbs, Paul Martin, Jane Robinson, Daniel Sagalowicz, and Paul Martin. "TEAM: A Transportable Natural Language System", Technical Note 263, SRI International International, Menlo Park, April, 1982.

[HARR83] Harris, Larry. "Artificial Intelligence Corporation", In: Sondheimer, Norman (ed), Tutorial on Natural Language Interfaces, Conference on Applied Natural Language Processing, Santa Monica, 1983.

[STER85] Stern, Rob, Bruce Anderson, and Craig Thompson, "A Menu-Based Natural Language Interface to a Large Database", NAECON: National Aerospace and Electronics Conference, Dayton, Ohio, May 20-24, 1985.

[TB&F83] Thompson, Bozena H and Fred B Thompson. "Introducing ASK, A Simple Knowledgeable System", Conference on Applied Natural Language Processing, Santa Monica, 1983.

[TENN80] Tennant, Harry R. "Evaluation of Natural Language Processors", Phd Dissertation, Department of Computer Science, University of Illinois, Urbana, Illinois, November, 1980.

[TENN83a] Tennant, Harry R, Kenneth M Ross, Richard M Saenz, Craig W Thompson, and James R Miller. "Menu-Based Natural Language Understanding", Proceedings of the 21st ACL, MIT, June, 1983.

[TENN83b] Tennant, Harry R, Kenneth M Ross, and Craig W Thompson. "Usable Natural Language Interfaces Through Menu-Based Natural Language Understanding", Proceedings of the Conference on Human Factors in Computing Systems, Boston, Mass, December, 1983.

[THOM83] Thompson, Craig W, Harry R Tennant, Kenneth M Ross, and Richard M Saenz. "Building Usable Menu-Based Natural Language Interfaces to Databases", Proceedings of the 9th VLDB Conference, Florence, Italy, October, 1983.

[THOM84a] Thompson, Craig. "Constraints on the Design of 'Menu-Based Natural Language' Interfaces", AI/CSL Technical Report # 84-03, March, 1984.

[THOM84b] Thompson, Craig. "Beyond Retrieval: Updating a Database using Menu-Based Natural Language Understanding", Proceedings of the 1984 Conference on Intelligent Systems and Machines, Oakland University, Rochester, MI, April 24-25, 1984.

[THOM84c] Thompson, Craig. "Recognizing Values in Queries and Commands in a Natural Language Interface to Databases". First Conference on AI Applications, Denver, December 5-7, 1984.

[THOM85a] Thompson, Craig, John Kolts, and Kenneth Ross. "A Toolkit for Building Menu-Based Natural Language Interfaces", 1985 ACM Annual Conference, Denver, Colorado, October 14-16, 1985.

[THOM85b] Thompson, Craig and Steve Martin. "Asking Spatial and Graphical Queries Using a Menu-Based Natural Language Interface", 1985 ACM Annual Conference, Denver, Colorado, October 14-16, 1985.

[THOM86] Thompson, Craig and Steve Martin. "Using Menu-Based Natural Language to Query an Integrated Database Management and Information Retrieval System", submitted to: The Second International Conference on Data Engineering, Los Angeles, February 4-6, 1986.

CALL FOR PAPERS

International Conference on

DATABASE THEORY

Rome, September 8-10, 1986

The Conference is intended to provide a European forum for the international research community working on theoretical issues related to database systems. It will be held in downtown Rome, in the main building of CNR (the Italian Research Council), and will be jointly organized by Istituto di Analisi dei Sistemi ed Informatica (IASI-CNR), Consorzio per la Ricerca e le Applicazioni in Informatica (CRAI) and Dipartimento di Informatica e Sistemistica, Università' di Roma "La Sapienza".

TOPICS

Major themes to be covered are (this is not meant to be an exclusive list): relational theory, logic and databases, conceptual models, knowledge representation and databases, deductive databases, theory of distributed databases and concurrency control, analysis and design of data structures, database interfaces, query processing.

PAPERS

Intended authors should submit six copies of a full draft paper before March 15, 1986, to the Chairman of the Program Committee:

Giorgio Ausiello
Dipartimento di Informatica e Sistemistica
Università' di Roma "La Sapienza"
Via Eudossiana 18
00184 Roma Italy

Authors will be notified of the program committee decision on their papers by June 1, 1986. Final versions will be due July 15, 1986. Proceedings will be available at the Conference in the form of preprints, and will be published in hardcover book a few months later.

PROGRAM COMMITTEE

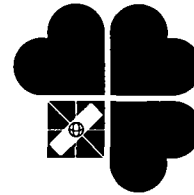
S.Abiteboul (France); G.Ausiello (Italy), chairman; F.Bancilhon (France, USA); A.D'Atri (Italy); W.Lipski† (Poland, France); M.Moscarini (Italy); J.Mylopoulos (Canada); J-M.Nicolas (France, West Germany); J.Nievergelt (Switzerland); C.H.Papadimitriou (Greece, USA); J.Paredaens (Belgium); D.Sacca' (Italy); N.Spyratos (France); J.D.Ullman (USA); M.Y.Vardi (USA).

ORGANIZING COMMITTEE

P.Atzeni (IASI-CNR), chairman; G.Ausiello (Università' di Roma); M.Moscarini (IASI-CNR); D.Sacca' (CRAI).

STUDENT AWARD

A 500\$ prize for the best paper authored solely by students will be awarded to commemorate our friend and colleague Witold Lipski (1949-1985), who had accepted to be a member of the Program Committee. Those who wish to be considered for the prize are invited to formally state their student condition in a letter accompanying the paper.



IFIP CONGRESS '86

Dublin Ireland 1-5 September 1986

10th WORLD COMPUTER CONGRESS

PROGRAMME COMMITTEE

Chairman

Prof. Dines Bjørner,
Department of Computing Science,
Technical University of Denmark, 343,
DK 2800 Lyngby, Denmark.
Telex: 37704 ddc dk
Telephone: + 45-2-872622 or + 45-2-881566.

Past Chairman

Prof. Denis C. Tsichritzis,
Institute of Computer Science,
Cretan Research Center,
P.O. Box 527, Heraklio Crete, Greece.
Telex: 262389 CCI GR.
Telephone: + 30-81-221171 or 225976.

Theoretical Computer Science

Prof. Vadim Kotov,
Computing Center of the Siberian Division
of the USSR Academy of Sciences,
Novosibirsk, 630090 U.S.S.R.
Telephone: USSR (383) 2655652.

Programming Science and Methodology

Prof. Dr. Ugo Montanari,
Dipartimento di Informatica,
Università di Pisa, Corso Italia 40,
I-56100 Pisa, Italy
Telex: 500371 cnuce i
Telephone: + 39-50-26362 or
+ 39-50-40864.

Software Engineering

Dr. Bálint Dömölki,
SZKI,
Institute for Co-ordination of
Computer Techniques,
1, Donati u. 35-45,
H-1015 Budapest, Hungary,
Telex: 225381 szki h
Telephone: + 36-1-868-632.

Mr. Horst Hünke,

Commission of the
European Communities,
Information Technologies and
Telecommunications Task Force,
Rue de la Loi 200,
B-1049 Bruxelles, Belgium.
Telex: 21877 comeu b
Telephone: + 32-2-235-7666.

Computer Engineering

Prof. Dr. Ryoichi Mori,
Institute of Information
Sciences and Electronics,
Tsukuba University,
Ibarakiken 305, Japan,
Telex: 3652580 untoku j.

Artificial Intelligence

Dr. Hervé Gallaire, Director,
European Computer-Industry Research
Centre GmbH, Arabellastrasse 17,
D-8000 München 81,
Federal Republic of Germany.
Telex: 5216910 ecrc d
Telephone: + 49-89-92699100.

Information Systems

Prof. Dr. Antonio L. Furtado,
Pontifícia Universidade Católica do
Rio de Janeiro,
Departamento de Informática,
Rua Marquês de São Vicente 225,
CEP, 22453 Rio de Janeiro, Brasil.
Telex: 102131048
Telephone: + 55-21-2744449

Distributed Systems

Prof. A. Danthine,
Systèmes et Automatique,
Inst d'électricité Montefiore B 28,
Université de Liège au Sart Tilman,
B-4000 Liège, Belgium
Telex: 41797 saunig b
Telephone: + 32-41-562691.

Computer Integrated Manufacturing (CIM-CAD/CAM)

Dr. J. Vlietstra,
A. T. & T. en Phillips Telecommunicatie,
P.O. Box 1168, NL-1200 BD Hilversum,
The Netherlands,
Telex: 43403
Telephone: + 31-35-892880.

New Informatics Applications

Prof. Anthony J. Wasserman,
Medical Information Science,
University of California, San Francisco,
San Francisco, California 94143, U.S.A.
Telex: 184 967 msg snds sfo ide
Telephone: + 1-415-666 2951.

Informatics in a Developing World

Prof. R. Narasimhan,
Tata Institute of Fundamental Research,
Homi Bhabha Road, Colaba,
Bombay, 400 005, India.
Telex: 11 3009 tifr in
Telephone: India (22) 219111.

Editor

Mr. Hans-Jürgen Kugler,
University of Dublin, Trinity College,
Department of Computer Science,
Dublin 2, Ireland.
Telex: 25442 tcd ei
Telephone: + 353-1-772941.

Organising Committee Liaison

Prof. John G. Byrne,
University of Dublin, Trinity College,
Department of Computer Science,
Dublin 2, Ireland.
Telex: 25442 tcd ei
Telephone: + 353-1-772941.

Congress Secretariat
IFIP Congress 86
44 Northumberland Road
Dublin, Ireland.

Telephone: 01-688244.
Telex: 31098
Telgrams:
Congrex, Dublin

Please reply to:

You should plan to submit a paper to the IFIP World
Computer Congress '86 - September 1-5, 1986 in Dublin.
The motto of the Congress, coined by the program
committee chairman is

INFORMATICS, A NEW AWARENESS

Awareness has to do with effective transfer of ideas
among

- edp professionals
- application systems designers
- computation scientists and engineers
- policy makers and planners

Information Systems (which includes data bases) is one
of the 10 areas of the Congress, having been described
as follows:

"This area is concerned with the creation, use and
maintenance of information systems in general.

Organisations show a growing need for information to
perform all kinds of tasks, ranging from routine
execution to decision making. Certain applications
require specific solutions. This is a source of
motivation for research in this area, and for
interaction with other areas. Examples include
distributed, personal, engineering, temporal and
inferential data bases, as well as office automation
and decision support systems.

Items for presentation include:

- Analysis and design of information systems.
- Theories and models for information, data and
functions.
- Education and training in information systems.
- Data base technology: hardware and software.
- Data administration.

There has been considerable technical progress in the
area, yet the interaction between research and practice
is far from satisfactory. Advanced techniques and
methods are not widely used or even known in many
organisations throughout the world. Research workers
often ignore what are the real problems faced by
practitioners.

Accordingly, new results will be presented whenever
possible in the context of the problems they aim to
solve and will be compared with other related efforts.
Tutorials and discussions will give ample opportunity
for critical evaluation of the state of the art and
identification of useful lines of future research."

Several highly reputed researchers have already agreed
to collaborate, presenting papers and participating in
panels.

The deadline for submission is November 1st, 1985. If
you plan to submit a paper, please write to me at once
indicating its tentative title. I will send you a copy
of the call for papers.

I hope to see you in Dublin!

A. L. Furtado

Take advantage of these pre-publication prices on

FORTHCOMING CONFERENCE PROCEEDINGS

Offered by the



IEEE Computer Society

- Proceedings: International Symposium on New Directions in Computing**
This symposium is being held in Norway on August 12, 1985. The proceedings reflects a broad scope of the state-of-the-art developments in computer systems and applications. Major areas include: CAD/CAM and graphics, distributed processing, software development methods, database, information storage and retrieval, artificial intelligence and robotics, and software tools and methods.
CQ639 (ISBN 0-8186-0639-8): August 1985, 406 pp.,
PRE-PUBLICATION list price \$40.50/member price \$20.25*
Estimated publication list price \$45.00/member price \$22.50
- Proceedings: 1985 International Conference of Parallel Processing**
This year's conference, being held in Illinois in August, gives birth to a proceedings that is clearly 50% larger than previous years. Some of this year's major topics consist of: algorithms, vector/array processing, problem mapping and scheduling, architectures, systolic systems, logic programming, operating system problems, computation, languages, memory, numeric processing, interconnection networks, data flow, numeric computing, network performance, and performance measurement.
CQ637 (ISBN 0-8186-0637-1): August 1985, 888 pp.,
PRE-PUBLICATION list price \$72.00/member price \$36.00*
Estimated publication list price \$80.00/member price \$40.00
- Proceedings: Third International Workshop on Software Specification and Design**
Being held in London in August 1985, this workshop's purpose is to identify major trends and areas of common interests in the formulation of requirements, specification and design of computer software.
CQ638 (ISBN 0-8186-0638-X): August 1985, 270 pp.,
PRE-PUBLICATION list price \$39.60/member price \$19.80*
Estimated publication list price \$44.00/member price \$22.00
- Proceedings: Eighth International Conference on Software Engineering**
This conference, being held in London in August, completes a decade of activity. The proceedings is divided into three streams: software process and environments, software engineering methods and metrics and their relation to the software process, and software engineering management issues.
CQ620 (ISBN 0-8186-0620-7): August 1985, c. 420 pp.,
PRE-PUBLICATION list price \$45.00/member price \$22.50*
Estimated publication list price \$50.00/member price \$25.00
- Proceedings: COMPINT '85**
COMPINT '85 is dedicated to various aspects of computer aided technologies from industrial through managerial, education and scientific applications. This year it is being held in Montreal, Canada, September 9-12.
CQ621 (ISBN 0-8186-0621-5): September 1985, c. 800 pp.,
PRE-PUBLICATION list price \$63.00/member price \$31.50*
Estimated publication list price \$70.00/member price \$35.00
- Proceedings: 1985 International Conference on Computer Design: VLSI in Computers**
ICCD '85 is a multi-disciplinary conference covering all aspects of design and implementation of VLSI computer and processor systems. ICCD '85 will be held in October in Port Chester, New York.
CQ642 (ISBN 0-8186-0642-8): October 1985, c. 800 pp.,
PRE-PUBLICATION list price \$64.80/member price \$32.40*
Estimated publication list price \$72.00/member price \$36.00
- Proceedings: COMPSAC '85**
The IEEE Computer Society's Ninth International Computer Software & Applications Conference is being held in Chicago in October. This year's topics include: software reliability/quality assurance, software engineering management/applications, emerging new software technologies, and expert/knowledge based systems.
CQ643 (ISBN 0-8186-0643-6): October 1985, c. 500 pp.,
PRE-PUBLICATION list price \$49.50/member price \$24.75*
Estimated publication list price \$55.00/member price \$27.50
- Proceedings: 1985 IEEE Symposium on Foundations of Computer Science**
The 26th Annual FOCS Symposium will be held in Portland, Oregon, on October 21-23, 1985. The proceedings combines original research on theoretical aspects of computer science.
CQ644 (ISBN 0-8186-0644-4): October 1985, c. 500 pp.,
PRE-PUBLICATION list price \$54.00/member price \$27.00*
Estimated publication list price \$60.00/member price \$30.00
- Proceedings: The 4th International Conference on Entity-Relationship Approach**
Concerned with both principles and pragmatics, this year's major theme is "the use of ER concept in knowledge representation". The fourth in the series, it is being held in Chicago in October.
CQ645 (ISBN 0-8186-0645-2): October 1985, c. 350 pp.,
PRE-PUBLICATION list price \$39.60/member price \$19.80*
Estimated publication list price \$44.00/member price \$22.00
- Proceedings: The Ninth Annual Symposium on Computer Applications in Medical Care**
SCAMC, now in its ninth year, will be held in Baltimore in November. This symposium offers subject areas designed to be used in all medical settings, for both health care professionals and computer professionals.
CQ647 (ISBN 0-8186-0647-9): November 1985, c. 1300 pp.,
PRE-PUBLICATION list price \$79.20/member price \$39.60*
Estimated publication list price \$88.00/member price \$44.00
- Proceedings: Conference on Software Maintenance 1985**
This 1985 Conference will be held November 11-13, in Washington D.C. Designed for software managers, maintainers, developers, and researchers, the proceedings deals with new solutions to the continuing challenge of software maintenance and maintainability, current advances, and current needs.
CQ648 (ISBN 0-8186-0648-7): November 1985, c. 300 pp.,
PRE-PUBLICATION list price \$39.60/member price \$19.80*
Estimated publication list price \$44.00/member price \$22.00
- Proceedings: International Test Conference 1985**
The conference, being held in November in Philadelphia, will be a technical forum on the testing of semiconductor components, boards and systems, with emphasis on the problems of testing increasingly complex LSI/VLSI/wafer scale devices and assemblies.
CQ641 (ISBN 0-8186-0641-X): November 1985, c. 900 pp.,
PRE-PUBLICATION list price \$72.00/member price \$36.00*
Estimated publication list price \$80.00/member price \$40.00
- Proceedings: Second Conference on Artificial Intelligence Applications**
This year's conference will be held in Florida, in December. The theme is engineering of knowledge-based systems and it will explore the technology, implementation, and impact of emerging application areas and will indicate future trends in available systems and required research.
CQ624 (ISBN 0-8186-0624-X): December 1985, c. 700 pp.,
PRE-PUBLICATION list price \$67.50/member price \$33.75*
Estimated publication list price \$75.00/member price \$37.50
- Proceedings: First International Conference on Supercomputing Systems**
The first conference of its kind, to be held in Florida, in December, will deal with the following aspects of supercomputing systems: hardware, software, artificial intelligence, and educational topics.
CQ654 (ISBN 0-8186-0654-1): December 1985, c. 600 pp.,
PRE-PUBLICATION list price \$59.40/member price \$29.70*
Estimated publication list price \$66.00/member price \$33.00

Orders for books to be published will be given priority fulfillment as soon as the title becomes available.

*NOTE: Take advantage of this special proceedings pre-publication price.

IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Fourthcoming TUTORIALS



offered by the
IEEE Computer Society

● Tutorial: Modern Design and Analysis of Discrete-Event Computer Simulations

by Edward J. Dudewicz and Zaven A. Karian

The objective of this tutorial is to provide a working understanding of the design, implementation, and analysis of computer simulations. The emphasis is on methods for application rather than on theory, and the goal is for the reader to be able to properly apply and interpret the design and analysis aspects covered in his/her own simulation studies of systems.

CONTENTS: Introduction; Random Number Generation and Testing of Random Number Generators; Sampling from Univariate and Multivariate Distributions; Efficient Design Techniques for the Choice and Reduction of Simulation Run Time; Fitting Distributions to Simulation Input/Output Data; Computer Sorting Methods and Their Use in Simulation; Applications of Simulations; Appendix—Review of Statistical Concepts and Modeling.

CQ597 (ISBN 0-8186-0597-9): August 1985, c. 500 pp.,
PRE-PUBLICATION list price \$40.50/member price \$32.40*
Est. post publication list price \$45.00/member price \$36.00

● Tutorial: Digital Image Processing and Analysis: Volume 2: Digital Image Analysis

by Rama Chellappa and Alexander A. Sawchuk

While Volume 1 (of this two volume set) deals with processing of images, Volume 2 deals with the analysis of images.

CONTENTS: Feature Extraction and Boundary Analysis; Region Analysis; Image Sequence Analysis; Multiresolution Image Analysis.

CQ666 (ISBN 0-8186-0666-5): October 1985, c. 780 pp.,
PRE-PUBLICATION list price \$59.40/member price \$32.95*
Est. post publication list price \$66.00/member price \$36.00

NOW AVAILABLE!

Tutorial: Digital Image Processing and Analysis: Volume 1: Digital Image Processing

publication list price \$66.00/member price \$36.00

Order the 2 volume set and receive a substantial 25% discount!

PRE-PUBLICATION list price \$99.00/member price \$54.00*
Est. post publication list price \$132.00/member price \$72.00

● Tutorial: Robotics (2nd Edition)

by C.S. George Lee, R.C. Gonzalez, and K.S. Fu

The latest edition of this popular tutorial gives an even better and more up to date summarization of the fundamental concepts and theories of robotics. Like the first edition, it includes concepts and theories at a mathematical level that requires a good background in vectors, matrices, kinematics, and dynamics of rigid bodies.

CONTENTS: Introduction; Robot Arm Kinematics; Robot Arm Dynamics; Planning of Manipulator Trajectories; Servo Control for Manipulators; Force Sensing and Control; Robot Vision System; Robot Programming Languages; Machine Intelligence and Robot Planning.

CQ658 (ISBN 0-8186-0658-4): September 1985, c. 630 pp.,
PRE-PUBLICATION list price \$40.95/member price \$26.95*
Est. post publication list price \$45.00/member price \$29.00

● Tutorial: Human Factors in Software Development (2nd Edition)

by Bill Curtis

In this latest edition, many new articles have been added, which will provide greater depth and appreciation for the impact of cognitive science on software engineering. This collection includes articles drawn from the following areas of psychological research on programming: cognitive ergonomics, cognitive psychology, and psycholinguistics. The areas covered range from language design, to team organization, to programmer selection.

CONTENTS: Introduction; Cognitive Models of Programming Knowledge; Learning to Program; Problem Solving and Design; Specification Formats; Programming Language Characteristics; Fault Diagnosis; Methodology; Epilogue: Future Directions in Programming.

CQ577 (ISBN 0-8186-0577-4): October 1985, c. 780 pp.,
PRE-PUBLICATION list price \$43.95/member price \$28.95*
Est. post publication list price \$48.00/member price \$32.00

● Tutorial: VLSI Testing and Validation Techniques

by Hassan Reghbati

This tutorial explores new computer-aided design and test tools that are continually being developed to cope with the ever increasing problems of VLSI complexity.

CONTENTS: An Introduction to Testing and VLSI Design; VLSI Design Validation; Failures, Fault Models, and Testing; Testable Design and Built-in Self-Test.

CQ668 (ISBN 0-8186-0668-1): August 1985, c. 450 pp.,
PRE-PUBLICATION list price \$35.10/member price \$26.10*
Est. post publication list price \$39.00/member price \$29.00

● Tutorial: Integrated Services Digital Networks

by William Stallings

The objective of this tutorial text is to provide a comprehensive introduction to the ISDN, dictated by concerns of breadth more than depth. The key concepts explored are: underlying technology, architecture, standards, and services.

CONTENTS: Overview; Standards and Regulations; Transmission Structure; User Access; Integrated Digital Network; Glossary; List of Acronyms; Annotated Bibliography.

CQ625 (ISBN 0-8186-0625-8): August 1985, c. 400 pp.,
PRE-PUBLICATION list price \$32.40/member price \$24.30*
Est. post publication list price \$36.00/member price \$27.00

Orders for books to be published will be given priority fulfillment as soon as the title becomes available.

*NOTE: Take advantage of this special tutorial pre-publication price.



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.

IEEE



Announcing the first IEEE Computer Society series on DATABASE SYSTEMS

These five books contain essential information on Database Systems. Order this unique package now, and for a limited time only, receive a savings of over 30%.

Reliable Distributed System Software by John A. Stankovic

The reader of this tutorial can expect to learn what reliability is, what reliability techniques are used in the different areas of distributed system software, and how reliability techniques can be better applied across all areas of distributed systems software especially in the distributed operating system area.

CONTENTS: Overview of Reliability (hardware and software): Overview of General Distributed Computer Systems Research; The Communication Subnet; Logical IPC and Distributed Programming Languages; Distributed Control; Structuring Distributed Systems for Reliability; Summary Collection of Software Reliability Techniques; Database Areas; Case Studies of Reliable Systems.

ISBN 0-8186-0570-7: July 1985, 400 pp., list price \$36.00

Database Engineering, Volume 3

This book binds together the four 1984 issues of the quarterly newsletter of the Technical Committee on Database Engineering. The issues feature such topics as: user interfaces, workstations and special purpose hardware, CAD/CAM systems, optical disks, spatial data management, comprehensive design environments now under development, early prototyping, and modeling transactions.

CONTENTS: A summarization of working group discussions at the second International Workshop on Statistical Databases; Engineering Data Management; Multimedia Data Management; Database Design Aids.

ISBN 0-8186-0672-X: February 1985, 262 pp., list price \$32.00

Distributed Database Management by J.A. Larson and S. Rahimi

This tutorial provides a thorough written description of the basic components of distributed database management systems, describes how each of these component works, and examines how these components relate to each other.

CONTENTS: Introduction; Transforming Database Commands; Semantic Integrity Constraints; Decomposing Requests; Concurrency and Replication Control; Distributed Execution Monitor; Communications Subsystem; Design of Distributed DBMSs; Case Studies; Glossary.

ISBN 0-8186-0575-8: January 1985, 678 pp., list price \$36.00

IEEE Computer Society Books— putting today's computer professionals in touch with tomorrow's technologies.

Recent Advances in Distributed Data Base Management by C. Mohan

By reading this text completely, the reader will be able to acquire a good understanding of the issues involved in DDBM. This tutorial assumes prior exposure to centralized data base management concepts and therefore is intended for systems designers and implementors, managers, data base administrators, students, researchers, and other technical personnel.

CONTENTS: Introduction; Distributed Data Base Systems Overview; Distributed Query Processing; Distributed Transaction Management; Distributed Algorithm Analysis; Annotated Bibliography.

ISBN 0-8186-0571-5: December 1984, 350 pp., list price \$36.00

Data Base Management in the 1980's

by James A. Larson and Harvey A. Freeman

This tutorial addresses the kinds of data base management systems (DBMS) that will be available through this decade. Interfaces available to various classes of users are described, including self-contained query languages and graphical displays. Techniques available to data base administrators to design both logical and practical DBMS architectures are reviewed, as are data base computers and other hardware specifically designed to accelerate database management functions.

CONTENTS: Introduction; Tools for Data Base Access; Coupling A Programming Language to a Data Base; Data Base Design; Data Base Management System Design; Hardware Aids.

ISBN 0-8186-0369-0: September 1981, 472 pp., list price \$27.00

TO ORDER: Return this form with remittance to:

IEEE Computer Society Order Department
P.O. Box 80452
Worldway Postal Center
Los Angeles, CA 90080 USA

YES, please send _____ set(s) of order #DDS14, the Database Series at this limited time offer of \$117.00 (\$50.00 off the list price) plus \$10.00 shipping charge.
California residents please add 6% sales tax.
Foreign orders must be prepaid.

Sorry, no substitutes or returns.

check enclosed Visa MasterCard American Express

card no. _____ exp. date _____

signature _____

name _____

affiliation _____

address _____

city _____ state _____ zip _____

country _____

phone/telex no. _____

purchase order no. _____

CQ



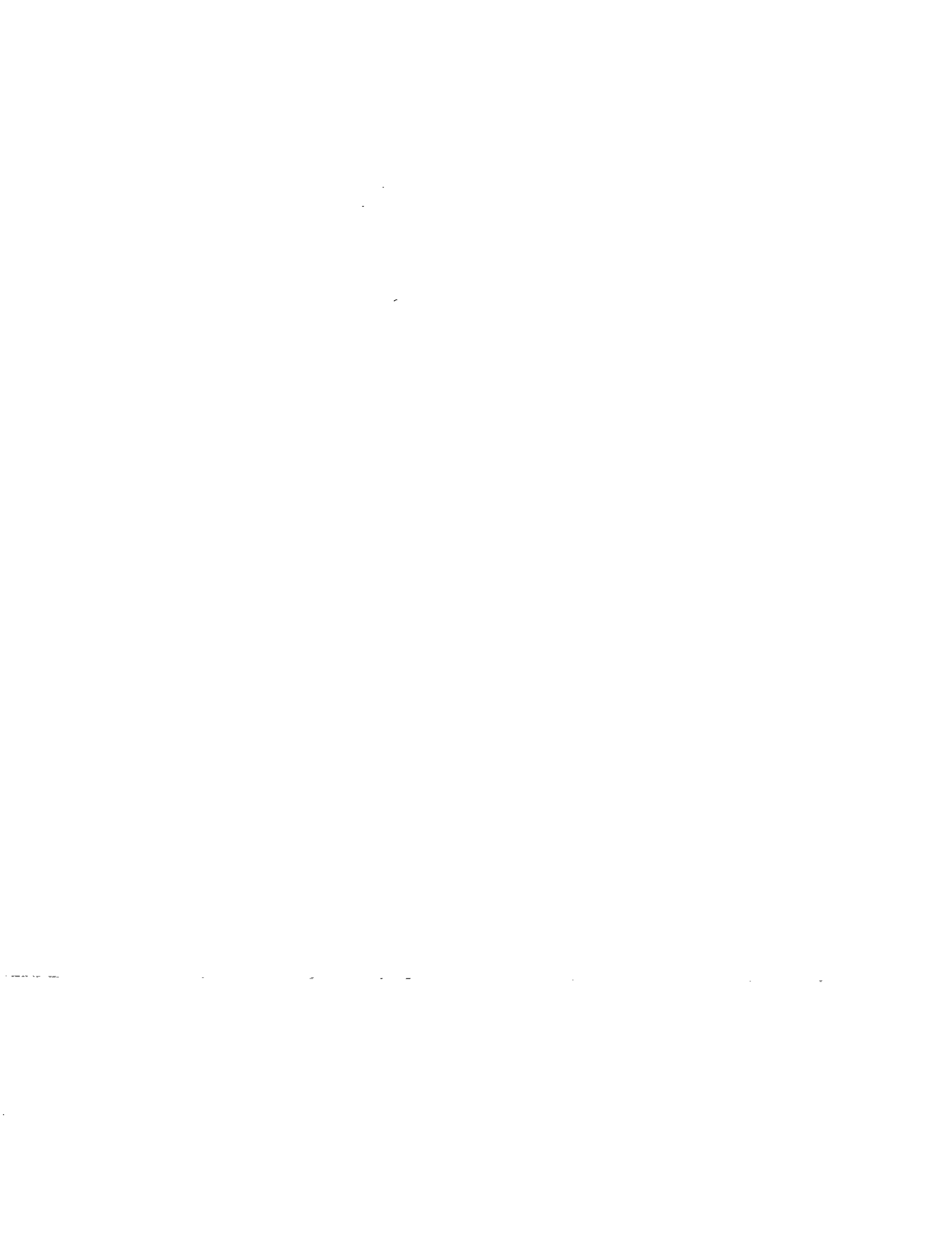
IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.

IEEE





PUBLICATIONS ORDER FORM

Return with remittance to:
IEEE Computer Society Order Department
P.O. Box 80452
Worldway Postal Center
Los Angeles, CA 90080 U.S.A.



Discounts, Orders, and Shipping Policies:

Member discounts apply on the **FIRST COPY OF A MULTIPLE-COPY ORDER** (for the same title) **ONLY!** Additional copies are sold at list price.

Priority shipping in U.S. or Canada, **ADD \$5.00 PER BOOK ORDERED.** Airmail service to Mexico and Foreign countries, **ADD \$15.00 PER BOOK ORDERED.**

Requests for refunds/returns honored for 60 days from date of shipment (90 days for overseas).

ALL PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE. **ALL BOOKS SUBJECT TO AVAILABILITY ON DATE OF PAYMENT.**

ALL FOREIGN/OVERSEAS ORDERS MUST BE PREPAID.

Minimum credit card charges (excluding postage and handling), **\$15.00.**

Service charge for checks returned or expired credit cards, **\$10.00.**

PAYMENTS MUST BE MADE IN U.S. FUNDS ONLY. **DRAWN ON A U.S. BANK.** UNESCO coupons, International money orders, travelers checks are accepted. **PLEASE DO NOT SEND CASH.**

ORDER HANDLING CHARGES (based on the \$ value of your order—not including sales tax and postage)	
For orders totaling:	Add:
\$ 1.00 to \$ 10.00	\$ 3.00 handling charge
\$ 10.01 to \$ 25.00	\$ 4.00 handling charge
\$ 25.01 to \$ 50.00	\$ 5.00 handling charge
\$ 50.01 to \$100.00	\$ 7.00 handling charge
\$100.01 to \$200.00	\$10.00 handling charge
over \$200.00	\$15.00 handling charge



PLEASE SHIP TO:

NAME

AFFILIATION (company or attention of)

ADDRESS (Line - 1)

ADDRESS (Line - 2)

CITY/STATE/ZIP-CODE

COUNTRY

IEEE/COMPUTER SOCIETY MEMBER NUMBER (required for discount)

PHONE/TELEX NUMBER

PURCHASE ORDER NUMBER

AUTHORIZED SIGNATURE _____

QTY	ORDER NO.	TITLE/DESCRIPTION	M/NM PRICE	AMOUNT

If your selection is no longer in print, will you accept microfiche at the same price?
 Yes No

CALIFORNIA RESIDENTS ADD 6% SALES TAX
 HANDLING CHARGE (BASED ON SUB-TOTAL) \$ _____
 OPTIONAL PRIORITY SHIPPING CHARGE \$ _____
SUB TOTAL \$ _____
TOTAL \$ _____

METHOD OF PAYMENT (CHECK ONE)
 CHECK ENCL. VISA MASTERCARD AMERICAN EXPRESS

CHARGE CARD NUMBER EXPIRATION DATE

 SIGNATURE CQ



Non-profit
Organization
U.S. Postage
Paid
Silver Spring, MD
Permit No. 1398