International Institute for
Applied Systems Analysis
IIASA www.iiasa.ac.at

Proceedings of a Workshop on Natural Language for Interaction with Data Bases, January 10-14, 1977

Rahmstorf, G. and Ferguson, M.

IIASA Collaborative Paper
December 1978

PROCEEDINGS OF A WORKSHOP ON NATURAL LANGUAGE FOR INTERACTION WITH DATA BASES

January 10-14, 1977

G. Rahmstorf and M. Ferguson, editors

CP-78-9
October 1978

International Institute for Applied Systems Analysis
A-2361 Laxenburg, Austria

PREFACE

IIASA is pursuing international cooperative research in several fields of applied systems analysis such as energy development strategies, regional development, and water systems research. This cooperation is dependent on a high quality of communication and documentation. Tools and techniques such as computer networks, data bases, and high-level interactive languages have an increasing impact on the efficiency of scientific work.

The Workshop on "Natural Language for Interaction with Data Bases" was a forum for discussions on the advantages and limitations of natural language as a man-machine communication tool. A special feature of the Workshop was that many of the speakers took the occasion to demonstrate software systems they had developed. In order to help assess the appropriateness of the various systems for IIASA a small, rather simple relational data base on energy resources was supplied before the Workshop to those interested in demonstrations. Sample outputs from this data base are included in several papers.

SUMMARY

This Report is a collection of papers presented at the "Workshop on Natural Language for Interaction with Data Bases" held at IIASA in Laxenburg, Austria from January 10 to 14, 1977.

The papers describe the research and results in attempting to produce a viable, useful, and flexible interface to various systems in Europe (acronymically AQL, PLIDIS, USL, DONAU, DILOS, KAIFAS, etc.) and in North America (OWL, INGRES, and LIFER). Most of these interfaces present to the user the feeling of working in an environment of relatively free and forgiving syntax. This is in marked contrast to the rather rigid syntax required by most commercial data base systems in their natural (mostly English) query languages. In addition there are discussions of the categorization of the semantic relationships within some natural languages as an aid to both understanding and knowledge representation.

CONTENTS

Introduction

G. Rahmstorf and M. Ferguson

THE WORKSHOP FORMAT AND OBJECTIVES

The Workshop "Natural Language for Interaction with Data Bases" was a forum for discussions and demonstrations of systems attempting to use "natural" language as an interface tool to specific data bases. The artificial intelligence community along with linguists has been studying the automatic analysis of natural language for more than a decade. It is probably fair to say that the general language processor, for reasons that are discussed in this introduction and many papers in this volume, is still elusive. These difficulties have led to applications of natural language processors in limited contexts such as specific data bases where the chance of successfully demonstrating a natural language access is much greater. However, the desire in most cases is to produce a natural language system that has potential for very wide usage.

At the same time the natural language systems were being developed, there was tremendous activity in data base system analysis and design. Data bases were first interfaced with formal programming languages but it has recently become obvious that there is a need to provide service for a casual user who has little programming knowledge. Thus, quite independently of the artificial intelligence community, the data base designers have been developing and marketing interactive (query) data base access languages. In most cases these are quite rigidly structured but, it is claimed, quite easy to learn. This naturally has led to a certain level of disagreement between these communities as to the need and usefulness of natural language interfaces to data bases.

The Workshop, during which the following papers were discussed, was to bring together the two communities to determine the art and need in natural language processing especially with respect to data base interfacing, to compare various systems that actually have been implemented, and to discuss where natural language interfaces might prove useful and whether there were reasonable alternatives. The latter objectives were difficult to attain as the great majority of the Workshop participants were primarily interested in the various aspects of natural language processing. The applications, although both real and interesting, tended to be with data bases that were sufficiently small that the implementation of the data base was rather straightforward. Questions relating to the access to already existing, relatively

large data bases, and the matching of the information contained in the data base with that required by the natural language interface processor remain largely unanswered.

All the authors were requested to describe their approaches, systems, and problems with implementing computer based natural language understanding systems in the context of data base access. The invited papers represent the distillation of the thoughts of the participants on their systems and the problems of automatic analysis of natural language in general. A complete discussion of natural language and data base technology involves vast quantities of detail where the implementation considerations usually have strong impact on what one would like to consider general principles. In order that the reader not be deluged in detail most authors have naturally concentrated on those aspects that they consider both peculiar and important. This shows up in considerably different data world views and allows only a loose grouping of the papers in this volume.

In addition to the papers, several of the authors responded to IIASA's invitation to demonstrate their systems. To facilitate a comparison IIASA provided a small relational data base of energy resource data. Details of the 11 relations involved are given in the Appendix. Some of the authors also responded to IIASA's request to use the sample data base as examples in their papers. This allows some consistency in the comparison of approaches from paper to paper.

Figure 1 shows a very general information system consisting of a user interacting with a natural language processor eventually wishing to get to a data base. The system interprets the natural language input with respect to a "language world model" and then translates it to a data base access sequence being restricted to one of the views supported by the "data model" of the data base system. The data interrelations as described by the "data base access model" are the simplest of the world models in the system and its language can be looked upon as providing the primitives into which the information requests must be translated in order to obtain the correct data from the data base. A major difference in approach in the papers is the relative importance of each model and the degree to which the various models are assumed to be integrated both conceptually in reality and in their system implementation.

The first group of papers suggests that there is a "clear and clean interface" (H. Lehmann) between the data base and the natural language processor. Thus the data base has a formal well defined access language and associated data model. Most systems of this group include a data base component that is either a standard software product or an experimental data base developed before and independently of the natural language processor (see, for example, H. Lehmann, M. King et al.).

The second group of papers tends to integrate the data and language models into one powerful structure the main purpose of which is representing knowledge in as wide a context as possible.

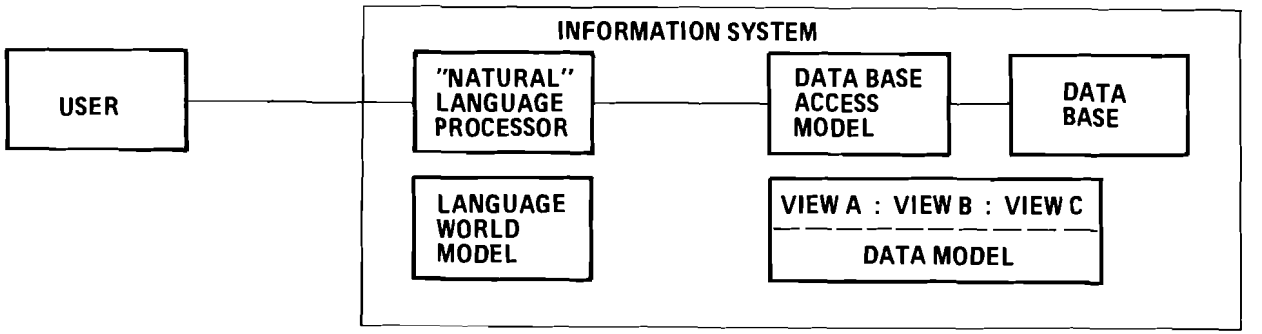


Figure 1. A general block diagram for a natural language interface to a data base system.

The interaction with the data base is usually viewed as one possible application of such a system. The authors in this group usually come from the artificial intelligence community and favor complex structures such as semantic networks as the basis of their language models.

The third group of papers is concerned with some specific problems of natural language analysis whose solution would greatly enhance the ease of automatic natural language processing. It is noteworthy that, although most of the difficulties exist in many natural languages, some of the troublesome points raised by the authors in this group are peculiarly vexing in only some languages.

The last group of papers discuss selected problems in the conceptual design of natural language systems and attempts to look for guidance in the relation between natural language processing and application fields such as information retrieval systems or robot controlling.

AN INTRODUCTION TO THE WORKSHOP PAPERS

Although each paper presented has a particular view and emphasis, they all attempt to deal with many aspects of natural language understanding and the data base interface. The purpose of this part of the introduction is to discuss the contributions of the papers with respect to the scheme indicated in Figure 2. This should allow the interested reader to access via subject the positions expressed in most of the papers.

The Natural Language Interface

Advantages

One of the major advantages of a natural language interface is its availability to a large number of users without (much) special training or the learning of a formal programming language. "A user who is not a computer specialist finds input in a formal language, however well designed, sufficiently repugnant to discourage him from using the system. The amount of effort required to develop a large data base system is only worthwhile if the resulting system can be used by a wide variety of nonspecialist users, who must therefore be specially catered for by the provision of facilities that will accept as wide a range of natural language input as possible " (King et al.).

A second major advantage is the potential universality of the natural language ability to express any kind of information --true or false, real or hypothetical, general or specific. Natural language is in a very real way the ultimate meta-language for any formal language. "Native speakers of English can usually communicate their knowledge of any domain of interest in English, perhaps augmented by specialized notations and vocabularies particular to their domain" (Szolovits). Guida argues that a natural

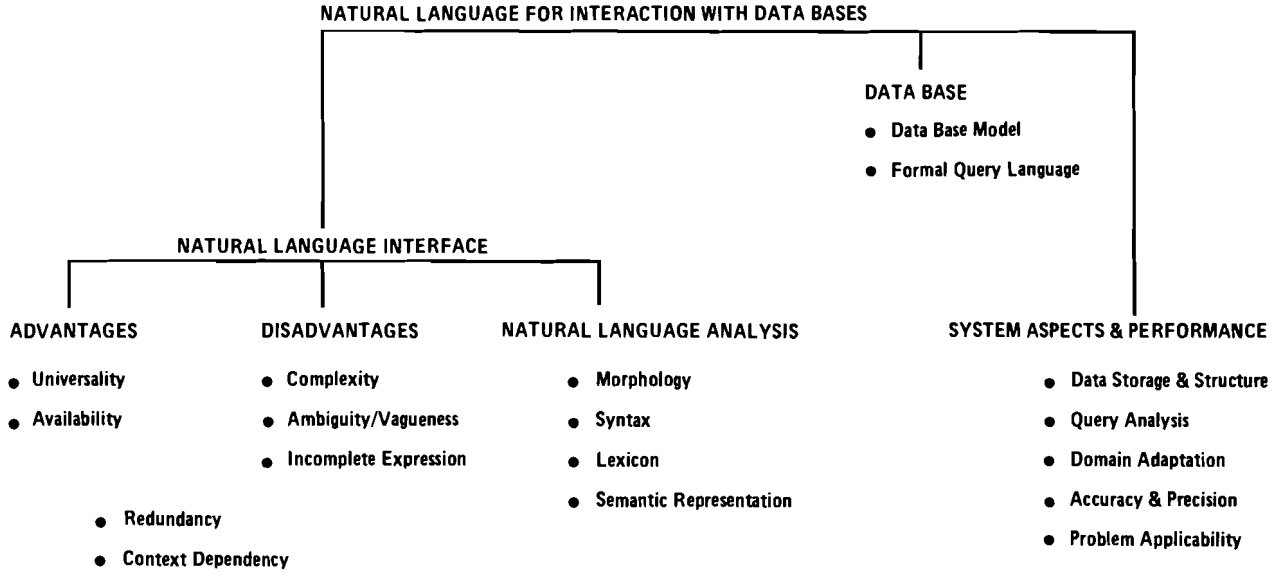


Figure 2. Introductory subject structure.

language will allow a user to start work on a problem in spite of uncertainties about the content and use of the system. A two way natural language dialogue would allow a user to start a terminal session and learn the operative characteristics of the system. Although two way natural language dialogue is considerably more difficult than just understanding, Szolovits specifically structures his knowledge representation to facilitate such a dialogue.

Disadvantages

The fact that natural languages have the advantages indicated in the previous section almost seems to require that they have a rather large list of disadvantages when trying to implement them in the rather unforgiving environment of a computer system. "The lack of an adequately developed linguistic theory and of a formalized corpus of knowledge are the main obstacles for building a system with general natural language understanding capability today" (E. Lehmann). The subtlety and range of expression possible in natural languages lead to syntactic and semantic structures that are far more complex than any formal language with consequent computer speed, efficiency, and system problems. Although questions of this type were not of paramount importance to most of the authors, most likely because of the relative immaturity of the entire field, Hendrix does have some notes on parsing speeds for his LIFER system.

One of the most difficult problems in natural language understanding is coping with ambiguity and vagueness. Examples in the general problem of natural language understanding abound and are mentioned in several papers. King et al. discuss the following example:

> Give me the name of the employee who works in the personnel department whose salary is more than 3000 francs a month.

"It is quite obvious to any person reading this that the 'whose' refers to the employee and not the personnel department. Yet there is no syntactic rule [in English] that could be used to determine this" (King et al.). Another example is given by H. Lehmann:

> Manager of Bill Jones.
> Manager of IBM.

In English, it requires a rather deep semantic analysis of sentences of this type to resolve the ambiguity. The ability to distinguish classes of this type could be used as a test of the semantic depth of the processor.

Virtually all ambiguities are resolved by means of semantic relationships. These relationships may be supplied by a common sense interpretation of the world or by a narrower context implied by the target subject. Krägeloh points to the examples below as being resolved by common sense:

John jumps higher than Peter.
John jumps higher than the Eiffel Tower.

Hendrix builds the context into his syntactic analyzer to handle the elliptical (incomplete) inputs as shown:

WHAT IS THE DEPTH OF THE GOLDEN SPIKE DEPOSIT?

then the input:

OF BELL CREEK?

The latter input will be interpreted as

WHAT IS THE DEPTH OF THE BELL CREEK DEPOSIT?

Most authors find it necessary to embed semantic information in their syntactic analyzers to direct the parsing (H. Lehmann, G. Hendrix) while Szolovits argues that in a fundamental sense all parsing is semantically based.

Some of the restrictions simplifying a natural language interface induced by a data base system are discussed by Krägeloh. He comes to the interesting conclusion that "relative clauses could be eventually excluded from the query language. Instead, the possibility of references to queries stated before is needed involving the well known problem of pronouns".

Redundancy in any system is capable of being either helpful or harmful. In ordinary conversation the redundancy in natural language serves with context and common sense as an error and ambiguity resolver. However the mechanisms of this are either not well understood or too difficult to implement (or both) and hence redundancy is more of a problem than a help in automatic natural language understanding. It is difficult to drop irrelevancies. Most formal languages strive for compactness, conciseness, and precision. Natural language seems deficient in all three. This normally leads to an augmentation of several natural language systems, for example USL (H. Lehmann) with arithmetic operators.

As a matter of fact, the differences between algorithmic processes and natural languages seem so severe that it has been suggested that there are fundamental nonalgorithmic aspects of natural languages that make the implementation of a true natural language interface impossible.

Natural Language Analysis

The automatic analysis of natural language reflects the traditional components of language namely, morphology, lexicon, syntax, and semantics. As mentioned earlier some natural languages present problems that are uniquely severe. The inflection system in German is the most important source of morphological problems in that language (see Krägeloh).

Syntax analysis is probably the best developed part of natural language analysis. Most authors prefer Wood's Augmented Transition Network (ATN) as the basis of their parser. "This technique was chosen as it seemed to be the best studied of parsing techniques and at the same time it may be handled easily by linguists without any special training in programming. The advantage of the ATN is its open-endedness, allowing the definition of new arcs, tests, and actions as required for the analysis of special natural languages" (Berry-Rogghe and Wulz). There are some example diagrams of an ATN parser in E. Lehmann while Berry-Rogghe and Wulz discuss the syntactic capacity of their parser with an example showing the actual and desired phrase structure.

The beginning of any language analysis is a recognition of the words or symbols that form the basis of the language. The words, along with information regarding their usage are carried in a lexicon or dictionary. Some authors (Briabrin and Senin) claim that different applications are only distinguished by their lexicon and not necessarily in ways involving different syntax or semantic structures. They distinguish between a "general context" independent of the data base and primarily syntactical, and a "local context" dependent on the information in the data base. A particular vocabulary of their system serves as a "physical embodiment of some local context". The syntax rules must be changed only when the general context is changed. The information actually carried in the lexicon varies considerably from system to system. Berry-Rogghe and Wulz in fact have two lexicons, a morpho-syntactic lexicon containing word forms with information on tense, number, gender, etc. and a semantic lexicon containing information about a word's equivalent internal representation such as its "sort", and number and type of arguments for each predicate.

There is some disagreement as to the size required for a lexicon in specialized systems but most authors agree with H. Lehmann that the miniworld represented by the data base needs only those words which express the contents of the files. PLIDIS has 10,000 nonlemmetized entries, King et al. has 4000 items, Krägeloh in KAIFAS plans to use 50,000 pharmaceutical terms, and

E. Lehmann has a core of 1500 German words deemed most important. Lehmann's dictionary contains mainly the structural or functional words (articles, pronouns, prepositions, conjunctions, and some adverbs) along with the most important content words.

In contrast to the above systems, which are trying to develop lexicons for specific cases, Guckler presented a system that was entirely lexicon. This was developed from a large German dictionary edited by G. Wahrig and implemented using the IBM information retrieval system STAIRS. It contains 17,000 German words with 50,000 units of meaning and associated lexicographic information. One of the most interesting objectives of this project is the identification of the very general concepts that are used in word definitions but cannot be defined by other concepts. These very general concepts are referred to in many of the papers by such names as semantic primitives, sorts, semantic markers, or basic concepts. In addition to the identification of basic concepts, it is hoped that the data base will help identify basic relations between concepts. This set of relations will also be of interest for the further development of information retrieval systems (see Rahmstorf).

Historically, most linguists, according to several language theories have distinguished sharply between semantics and syntax. Most contributions understand syntax to mean a system of rules defining how language expressions can be generated by such classical word and grammar categories as noun phrase, verb phrase, noun, adjective, and terminal symbols such as "house", "computer", or "good". Several systems (Krägeloh, H. Lehmann) use context free grammar rules to define the language syntax. However it has been long noted that many language expressions generated according to just the syntactic rules and categories lead to meaningless expressions. In addition, the syntactic structure (phrase structure diagrams) of a meaningful expression does not represent the logical relations between its meaningful components and does not represent synonymity or quasi-synonymity with other expressions of different syntactic structure that have the same or similar meaning. Thus new categories and structures based on word meanings and relationships have been introduced to restrict the range of expressions that can be generated to, hopefully, only meaningful expressions. These additional rules, relations, and concepts comprise the semantic part of the processor. Structures of meaning are called by such names as logical representations, conceptual networks, or semantic networks. Various abstractions and simplifications, with little common agreement, are made by various authors to limit the size of the semantic processor. The "template" analysis of some processors (Hendrix, H. Lehmann) are examples of systems that blur the historical semantic and syntactic distinction.

E. Lehmann, as a first example of semantic structuring, represents natural language statements by a hierarchical semantic network whose nodes are individual entities of different sorts (objects, locations, states, and intentions), concepts (classes, properties, relations, or functions of objects, events), numbers,

temporal relationships, quantities, and different quantified variables. The arcs of his network are basic semantic relations such as AG (agens), RECIP (receiver), CAUS (causality) and METH (method) and a good example of this type of semantic network is in his Figure 4.

A many sorted representation is also used by Krägeloh who calls his entities object types. Berry-Rogghe introduces a set of "sorts" to the usual symbols of the predicate calculus. Szolovits discusses several kinds of concept specialization. His sample memory taxonomy reflects the many-sortal approaches of several of the other authors.

H. Lehmann uses a limited set of role names for the domains of the relational structure of his data base structure, which are then related to deep structures in natural language queries. He describes how his system, USL, interprets such natural language structures as ⟨adjective⟩⟨noun⟩, ⟨noun⟩ of ⟨noun phrase⟩, and ⟨verb⟩⟨noun phrase⟩.

King et al. use an adaption of Wilk's preference semantics system where a verb, for example, may express a preference for an animate subject, or an adjective a preference for being a quality of a physical object. Such preferences are extensively used for semantic interpretation and reference determination. Their paper describes in greater detail an intermediate semantic representation and how it develops using a specific natural language sample statement in their system.

A very specific semantic problem is addressed by G. Lau. He attempts to categorize the 50 or so causal prepositions that exist in German. In so doing he distinguishes several different subcategories of causal relations. The subcategories can be added to the prepositions in the lexicon and can be used to disambiguate prepositional phrases.

Data Base

Most of the systems assume a version of Codd's relational data base as the conceptual data model. King et al. give a short and clear description of this model in their paper. The authors chose "this model because it is well developed theoretically, semantically complete, and logically transparent". Briabrin chose a hierarchical model and Krägeloh chose a set/entity model. Both use query languages matched to their choice of models. The only "complete" data base system was described by Stonebreaker with his relational data model based INGRES system. His paper indicates a very comfortable implementation on a PDP-11 and he details both the model and the debate over the Codd model in the past few years.

All of the natural language systems developed as interfaces for data base systems rely on an intermediate data base query language that is usually at a higher level than the path access

language required to actually find a data item. The formal language is represented in Briabrin's system DILOS by the "phi" language, in Krägeloh's system KAIFAS by a set language, in Stonebraker's system INGRES by QUEL, and in H. Lehmann's system USL by a formal DB language, and in King's system by the APL based AQL.

Stonebraker introduces a simplified graphical based language called CUPID and suggested that there was really no need or place for a natural language interface for a data base system. He felt that his language represented a very comfortable, easily learned, and maybe even forgiving human engineered interface.

System Aspects and Performance Evaluation

Bernorio et al. and Guida discuss some of the basic considerations that could influence the gross system design of a natural language system that could be used as an interface to many different application systems. Bernorio argues that the robot example in his paper could be a prototype for a data base system. Both papers argue that it is better to adapt other systems than to reinvent. They argue very strongly that a dialogue is absolutely necessary to produce a comfortable interface and suggest that there really is no intrinsic difficulty to achieve such a dialogue. They also suggest various structures that could be used to achieve these goals.

It is possible to evaluate the performance of the various systems presented according to a set of criteria that determine both the ease and universality of their applicability. Since the natural language processors must work with data in data bases and obtain information from the data, they must be concerned with

- the storage and structuring of data and information;
- query analysis, information retrieval, deduction and problem solving interfaced to the data base;
- adaptation to new data bases or domains;
- accuracy and precision of retrieved information; and
- problem applicability.

Most of the systems described, with the possible exception of INGRES (Stonebraker), have either minimal or no concern with either data input, storage, or structure. Several of the systems require that data be entered in their implementation language LISP (e.g. LIFER (Hendrix), DILOS (Briabrin and Senin)) or do not give sufficient information to determine entry procedures. The interfaces are primarily there to analyze input queries. H. Lehmann describes an on line update of the data base with declarative statements. The data base is generated by other techniques.

E. Lehmann's semantic network based system extracts information from input statements and adds it to the data base. An example of this is in the analysis of Victor's flight to Laxenburg. The text analysis would result in a richly connected semantic network, which is automatically updated and extended by each new text. This approach is applicable if the input text contains truth with respect to the semantic net.

The optimal balance between the generality of the natural language query analysis and the ability of the interface to use the information in the data base for ambiguity resolution is hard to obtain. If the interface uses sophisticated problem solvers during the query analysis then the data base must be structured compatibly with the problem solvers. This unfortunately is rather restrictive at the moment and would leave out most present data base structures. There thus is a dilemma that relates the power of the techniques used for query analysis, the universality of the interface, and the structuring of the data base. None of the papers suggest procedures for restructuring, either physically or logically, a presently constructed data base. However, this does not preclude the building of a data base in the required image. Moreover, there is an implicit assumption that the problems of data base maintenance such as data integrity and access control are either solved or belong to someone else.

If it is assumed that the data base structure is now compatible with the natural language interface and the problems mentioned above are solved there is still the question of whether a change in domain or application is easily accommodated. All of the systems described were developed for a specific application domain such as pharmaceutical drugs for KAIFAS (Krägeloh), environmental data for PLIDIS (Berry-Rogghe and Wulz), or robot control for DONAU (Bernorio et al.). However all systems were designed to "easily" change domain. USL (H. Lehmann) requires only a change of the lexicon by associating the new application domain terminology with the relations and relation domain names. For the more complex semantic network organized data bases such as that required for E. Lehmann's system the user will most likely not know in detail the organization of the information so that the move from a tested application domain to a new application domain will be more difficult. Briabrin claims that all that is required to change domain in DILOS is a change in vocabulary. It is difficult here, as in other cases, to evaluate whether the claims for easy adaption are justified. There is no agreed measure for ease of adaption.

The applicability of the interfaces and systems to new domains is also influenced by the linguistic coverage of the interface. Although all interfaces would like to have universal coverage, this is clearly an unattainable goal. This regonitiion results in a conscious restriction of the coverage to sublanguages such as kernel sentences only, noun phrases only (Rahmstorf), or limited vocabulary. Unfortunately, restrictions to a natural sublanguage are difficult to define and may be difficult to learn.

As a way around this some authors suggest allowing slightly ungrammatical sentences. Again though there is no accepted way of defining what is slightly and what is grossly ungrammatical.

The usefulness of a natural language interface is obviously influenced by the accuracy and precision of the information extracted from the data base. It is doubtful whether a natural language interface will ever give absolutely accurate and precise information. Although the input query is translated into a precise request to the data base, the inherent fuzziness of the natural language could easily result in two users making exactly the same request disagreeing on the precision of the outcome. The additional problem is that there is no way to measure precision for general data bases although partially subjective measures do exist for bibliographic data bases.

The determination of the applicability of natural language interfaces to real problems has not been answered in general at this Workshop. There was a distinct split between those participants that had spent most of their time building data base systems and those that had built natural language interfaces. The latter claimed that the natural language interface was necessary to attract the casual user while the former claimed that the simple highly structured query languages that are being marketed with data base systems are sufficient. What is lacking are any measures of user satisfaction of the various interfaces. In fact there has been very little real usage and none reported at this Workshop that would allow objective evaluation of any of the claims of the usefulness of natural language interfaces. What is clear from the Workshop is that the various systems are at the point where they can be introduced to real users for evaluation. The second clear point from the Workshop is that the data base technology and the natural language technology have not yet merged. The gap is precisely determined by the level of semantic based knowledge structure imposed or superimposed on the data base structure. With present technology, knowledge structure for large data bases would be quite expensive. A third point is that there are still a large number of problems in natural language analysis and that the solutions may turn out to be dependent on specific natural languages.

NATURAL LANGUAGE INTERFACING TO DATA BASE SYSTEMS

Access to a Data Base System Via Natural Language

K.-D. Krägeloh

GOALS OF NATURAL LANGUAGE PROCESSING FOR DATA BASE SYSTEMS

Simulation of Natural Language Understanding

Natural language communication with the computer has a relatively long tradition in informatics. Depending on their objectives, these activities fall into one of two disciplines, artificial intelligence (AI) or data base technology (DT).

In AI the conception of language understanding systems is part of the more comprehensive aim to simulate cognitive processes on the computer [1,2]. The basic concern of AI in this connection is the linguistic component of cognition, i.e. the assignment of meaning to language entities. Characteristic for man is his capability of producing a cognitive image of his environment. This image (called a model) is always an abstraction from the real world, chosen with respect to the purpose it is to serve. In language understanding, the statements about the real world are related to the cognitive model (assignment of meaning), and thus cause reactions such as modifications of the model, evaluation of the model, or answers.

The simulation of this process above all requires specification of a modeling system (MS), by means of which any environment or part of it can be described as a model (representation problem). A language understanding system is always based on a model formulated in some MS. Further, there is a need for mechanisms that put natural language expressions in relation to that model (fitting problem [1]).

Characteristic for human language understanding is the fact that meaning cannot directly be constructed from some basic units of meaning [1]. Instead, complex relations between the model objects enter as well into the meaning of language units such as words. Winograd mentions words like "virtue" or "democracy" as examples. In order to account for the aspect of relations, special MS are usually developed for language processing. This, however, leads to very complex MS and, consequently, to extensive models even in those cases where only small sections of the environment are considered (e.g. semantic memories [3], dependency networks [4], demons [5]).

Obviously the complexity and the size of the model affects the fitting problem and, if of concern, the rules for deriving the system reactions. Take as an example the task of finding

in an extensive semantic net all those subnets corresponding in their structure to a given net.

Both in human cognition and in the case of its simulation, models must be physically represented, and operators for manipulating them must be realized. In the first case we assume a practically unlimited memory (brain, neurons, etc.). This provides a fully associative solution of the fitting problem even for highly complex model structures, such as recollection of impressions, moods, etc. In computer simulation, however, we have to deal with physical devices and processes of limited capacity and duration. Given a complex MS and its corresponding extensive models, this merely allows for the description and representation of a drastically limited portion of the environment. Likewise, no manipulation of models on an associative basis is directly possible on a computer as yet. Instead, the associative behavior must be simulated thus causing considerable costs and process times, e.g. long response times in an interactive mode.

This problem will certainly become less troublesome as new hardware technologies become commercially available (use of microprocessors, new storage divices, LISP-machines, etc.). Still, it is by no means clear from current discussions whether human cognition and its simulation on computer really differ just quantitatively and not qualitatively [6].

If one approaches the same problem from the DT side, however, one must take into account the necessity of administrating and processing very large volumes of data. Consequently, even though a data base is usually again regarded as the model of some real world [7,8], the MS cannot be chosen completely at will but must meet a number of conditions:

- At justifiable costs, the MS must allow for the description of even such worlds whose number of objects and facts to be modeled is very large (i.e. model representation should take up as little storage space as possible).

- The rules for manipulating the models should be kept simple, since the time for evaluating even extensive models is very limited (acceptable response times in a data base system).

Under these conditions, data base systems must restrict their MS to those that can be formalized and are simple compared to those in AI. Models in DT will abstract from the real world far more than the usual AI models.

Consequently, the range of situations to which they are applicable will be much smaller than in AI, since they are much more likely to reject details that one may wish to include in them. Thus, models in DT are generally oriented towards a comparatively narrow purpose. Since in DT the operators defined on models are few, they are usually included in the MS which is then called a data model [9,10,11,12,13].

Natural language access to data base systems has frequently been discussed. By providing easy access to users not familiar with formal models one hopes the system will become more widely available [14,15,16,17].

Because of the differences in MS, it is obvious that language understanding systems for large data bases have to be based on something other than the simulation of cognitive processes. In DT the MS are rigorously defined by the interfaces of existing data base systems. Consequently, the language can be restricted to an extent necessary and reasonable for the usage of the system. Under these conditions natural language in data base systems, although much more diverse than programming languages, is still a formal language.

For practical use, these restrictions are quite acceptable. For instance, tests of user behavior in data base systems that provided (simulated) access by unrestricted English language have shown that the wealth of natural language expressions is never utilized [18]. Even more, certain query structures were used almost all of the time, thus indicating that a natural language interface for data base systems becomes highly stylized from the user's point of view, too. This might be due to a certain lethargy of the user, who finds it difficult to conceptualize complex statements [19].

The objective of this paper is to examine the consequences for natural language processing subject to the conditions imposed by data base system interfaces. This will include discussion of the question that results from the fact that AI may be incorporated in the development of natural languages for data base access, and to what extent, without violating the aspect of practicability. In addition to AI, text analysis techniques in documentation (automatic indexing, morphemic analysis [20,21,22]) are another large area that may contribute their results. The conclusions of these considerations will be illustrated in this paper by means of natural language access to a particular data base system. As mentioned above, the differences in MS have a clear effect on the linguistic approach. We shall explore this further by giving a rough outline of the development of language processing in AI and DT.

Approaches to Natural Language Processing in AI

The early language processing systems in AI concerned themselves with the problem of translating one natural language into another [23]. These systems possessed a certain knowledge of the external structure of the languages (grammar), and of the correspondences between words of the languages (dictionary of synonyms). These approaches failed, since grammar and dictionary alone proved insufficient to deal with the different meaning of sentences like [1]:

(1) The fish was bought by the cook.
(2) The fish was bought by the river.

The system would have to know something about the real world to which the sentence is applied in order to recognize the difference in the meaning of "by the cook" (person) and "by the river" (place).

Owing to the failure in the early sixties of these approaches in translation one turned to the broader question of how to simulate language understanding on the computer in general. One indication of a computer system's capability to comprehend natural language would be, for example, whether in man-machine communication the machine responds in a manner that seems "plausible" to its human partner. A system would be perfect if it passed the Turing test [24].

In order to enable the simulation of human language understanding, at least knowledge about the meaning of words relative to a specific environment must be provided (as we pointed out earlier). For this purpose the environment or part of it has to be modeled (MS). A certain portion of the model is assigned to the word as its meaning. The examples below, however, show that this approach is still not sufficient [25]:

(3) John jumps higher than Peter.
(4) John jumps higher than the Eiffel tower.

In order to recognize that (4) is meaningless, the system requires a more extensive knowledge of the environment than the (complex) meaning of individual words. The model must contain the fact that no living being can jump higher than the Eiffel tower, in general, it must establish relations between model parts.

Some of the MS used in the past cannot meet this condition or, at best, only if applied to sections of an environment too small to be of any interest. In particular, this applies to the earliest language understanding systems such as STUDENT [26], SIR [27], BASEBALL [28], ELIZA [29]. For example, the STUDENT system used equations as MS, i.e. it was restricted to a small mathematical world; ELIZA used patterns that were suitable only for very specific situations of a dialogue.

Many MS have the characteristics of a formal language; best-known among these is predicate logic (applied to a language understanding system by Coles [30]).

In a formal language MS the environment is ultimately modeled by means of certain basic units (axioms). Strictly speaking, they already belong to the class of MS that do not meet the demand for deriving the meaning of words from complex

structures with diverse dependencies between model parts. Winograd [31] or Woods [32], therefore, preferred to construct more complex MS from programs that provide both the description of environment situations and operational changes.

Other authors (e.g. Schank [4]) argue that language understanding could only be simulated on the basis of cognitive relations, and correspondingly developed as MS complex data structures such as semantic memories [3] or dependency networks [4]. The representation problem is solved by demonstrating the capabilities of the MS on severely restricted worlds (blockworld [31], airline guide [32], lunar geology [33], industrial enterprise [34]).

Approaches to Natural Language Processing in DT

The application of computers to the processing of large data sets in industry and public administration lead to the development of a class of special-purpose program systems known as "data base systems". A data base system provides for mechanisms for the description, management, and processing of large sets of data.

Modern data base systems [11,35,36,37,38] are based on MS (designated as "data models") that make feasible the modeling of extensive environments. The structure of most MS is such that it is possible to formalize the description of models according to comparatively simple rules and to express their processing by means of algorithms. The formal languages developed for that purpose are called the interfaces of data base systems. Well known MS in use are the relational model [9], the hierarchichal model [13], the network model [10], and the binary relation model [12].

As soon as one attempted to make data base systems available to the casual user unfamiliar with EDP, even simple formalization rules and their corresponding formal languages proved to be a severe handicap to many users. Natural language as an end user language offers some hope of overcoming these problems, since the user need not learn a new and artificial language, but will only have to observe restrictions on a language already well known to him. Natural language access to data base systems could be the particular advantage in areas like industrial management, medicine, pharmacy, engineering, or public administration where even in data retrieval--as part of a problem solving process--the user may employ the very language in which he generally formulates his problem and solution (see also [17,33]).

The scope of meanings of natural language queries to a data base system is determined by the MS (data model). Generally, such an MS is not oriented towards cognitive theories as it is in AI. The interface of the data base system is maintained no matter what language is chosen for accessing it, i.e. in the case of natural language access, too. In these systems the aim

of language processing is to translate natural language queries into expressions of the formal language at the system interface.

Kellog was one of the first to advance this approach. He chose as the MS of his CONVERSE system [39] a formal language dedicated to information retrieval and hence containing operators typical for this purpose. Likewise, Woods put his procedural semantics into the form of an interface in which predicate and function symbols were defined for the underlying procedures. Both these approaches fall into an area between the AI and DT methodology. Also to be included is the work by Thompson [40], whose ring structures may be interpreted both as the representation of cognitive relations and as the realization of a binary relational model [12]. While Thompson did not develop a formal interface, he--like Woods--was heavily concerned with the integration of large data bases into his REL system.

The expressiveness of natural language goes far beyond the power of the interfaces of available data base systems. In order to make some use of this expressiveness the formal MS must at least allow for lengthy expressions based on a small number of operators (such as the definition of algorithms, nesting of functions). In general, this capability is only found in so-called navigating systems [41]. In these a complex problem is described by a single expression divided into a large number of successive or parallel steps during processing. The results of steps serve as a guide to the following steps for further search in the data base. A classic example is the relational model; suggestions exist for a natural language access to it [17,42]. Counter examples seem to be the network model, the hierarchical model, or simply file management. Correspondingly, no approaches have been published for use of natural language access to them.

Other data models that might provide a basis for navigating systems are binary relations [12], LEAP-structures [43], or mathematical sets and relations [16]. The latter will serve as our starting point for demonstrating natural language access to a data base system.

A SET-THEORETIC MODELING SYSTEM (SET LANGUAGE)

The set-theoretic modeling system of KAIFAS is a formal language MS based on set and relation algebra. In order to express algorithms in the language, the language elements must be classified into operators, operands, and the control structure [38,44]. The operand (object) types of the set language together with their symbols are listed in Table 1 together with examples from a pharmaceutical application. The symbols for the instances of types are generated by indexing the symbols of the corresponding object types. The language includes the standard set-theoretic operators (Table 2). Special operators map relations to sets. Further, some logical and relational operators are defined.

Table 1. Object types and operators.

Object types	
I	Individuals, e.g. Thomapyrin, Perphyllon
M	Sets, e.g. drugs, diseases Lists of individuals
R	Relations, e.g. indication, contraindication, manufacturer Lists of pairs of individuals (Work is under way to cover n-ary relations)
Z	Numbers
D	Measures, e.g. 4 tablets/day
F	Measure functions, e.g. dosage Lists of ordered n-tuples whose last component is a measure
B	Truth values

Operands

$I_1, I_2, I_3, \ldots, I_n$, $M_1, M_2, M_3, \ldots, M_k$, $R_1, \ldots$

The control structure of the language determines the sequential order in which the operators are executed. This is indicated in the language by expressions in functional notation, e.g.:

$$\in(I_{\text{Steicardin}}, M\cap(M_{\text{prescription drug}}, Vg(R_{\text{drug}}, I_{\text{heart neurosis}})))$$

(Interpretation: Is Steicardin a prescription drug for heart neurosis?) The operators were applied in the sequence: Vg, M∩, ∈. Loops are introduced by the use of bounded quantifiers. They offer the (only) possibility to formulate queries of any degree of complexity:

(1) Are all drugs for glaucoma prescription drugs?
(2) Which antibiotics are incompatible with cytostatic drugs?

In both examples the flow of control is identical. For each element of a given set (i.e. "drugs for glaucoma" or "antibiotics) a certain condition (e.g. to be a prescription drug,

Table 2. Operators.

On sets:	
$Mb(I_1,\ldots,I_n)$	set construction
$M\cup(M_1,M_2)$	union
$M\cap(M_1,M_2)$	intersection
$Km(M_1,M_2)$	set difference
$Kz(M_1)$	cardinality
On relations:	
$Ko(R_1)$	converse
$Rb(R_1,M_1)$	restriction $(\{(x,y)\mid(x,y) \in R_1 \wedge x \in M_1\})$
$Rp(R_1,R_2)$	product
$R\cup(R_1,R_2)$	union
Reduction of binary relations:	
$Vo(R_1)$	domain $\{x\mid\exists y: \ (x,y) \in R_1\}$
$Na(R_1)$	range $\{x\mid\exists y: \ (y,x) \in R_1\}$
$Vg(R_1,I_1)$	individual domain $\{x\mid(x,I_1) \in R_1\}$
$Ng(R_1,I_1)$	individual range $\{x\mid(I_1,x) \in R_1\}$
Reduction of measure functions:	
$Fw(F_1,I_1)$	measure number
Logical operators:	
$\in(I_1,M_1)$	test on set membership
$\subset(M_1,M_2)$	test on set inclusion

to be incompatible with cytostatic drugs) is subject to a test. In (2) only those elements are listed for which the test yields "true". (1) corresponds to the following formulation in the set-theoretic machine:

$$AL(x_1, Vg(R_{drug}, I_{glaucoma}), \in(x_1, M_{prescription\ drug})) \quad .$$

Important quantifiers are:

AL:	all, every,	EI:	some,
DB:	which,	ZB:	how many.

Bounded quantifiers contain three arguments:

- The name of a bound variable, each of its substitutions defining an invocation of the loop: x_1.
- An expression resulting in a set of objects (range): $Vg(R_{drug}, I_{glaucoma})$.
- An expression for the condition resulting in a truth value (scope): $\in(x_1, M_{prescription\ drug})$.

Expressions containing quantifiers must be in prenex normal form, i.e. quantifiers must always appear as the left-most part of an expression.

PREMISES FOR LANGUAGE ANALYSIS IN DATA BASE SYSTEMS

The main purpose of data base systems is to provide tools for the management and retrieval of large volumes of data that are maintained on peripheral storage devices. Access by natural language can only be justified if it does not consume an inordinate amount of resources such as storage space or processing time. Consequently a number of restrictions must be imposed on a natural language interface in a data base system as compared to the interface of a general language understanding system.

Restrictions on the Natural Language Interface

The Natural Language Interface Should be Describable in Terms of a Simple Syntax Model

This suggests limiting the syntax model to context-free grammars. Previous research has shown that context-free grammars are inadequate for the purpose of defining natural language, but the examples used in the literature for demonstrating the need

of more complex grammars are of a rather exotic nature, at least as far as their applicability in data base systems goes. Indeed, Kratzer [45] defined a comparatively large subset of natural German by means of a context-free grammar without indicating any need for restricting the semantics of his subset. Therefore one would expect a context-free definition to be justified all the more in connection with formal MS and the restrictions of the semantics corresponding to it. The work by Malhotra [18] also indicates that there is no need for an extensive language definition in the data base area. Hence the application of context-free languages does not seem to place unreasonable constraints on the formulations a user may be able to use.

Simple Procedures Should Be Chosen for Morphemic Analysis

The analysis of natural language, and German in particular, introduces the problem of morphemic analysis [20,46]. Depending on the permissible error rate (incorrectly reduced word forms) costs and efforts for solving this problem may rise arbitrarily high (see, for example, [21]). Preferebly, simple procedures should be chosen here again, the error rate resulting from even very simple procedures (masking [47]) is surprisingly low (~30%).

Verbs Should Be Omitted from the Interface

Both these requirements discussed can be justified all the more, since, in defining a language for data base systems, verbs may be omitted to a large extent. Obviously, verbs are indispensable as soon as a MS accounts for temporal relationships and, consequently, permits the description of dynamic processes (see, for example, REL [40]). Data bases of that kind, however, have so far never gone beyond pilot studies; data base systems in practical use do not include them yet.

The Parser Should Be Simple and, on Average, Fast

Again, this requirement influences the complexity of the total system. In the literature a number of parsers for context-free languages are given [48,49,50]. Their efficiency is measured in terms of an upper limit for the time needed for processing sentences containing n words (in general the efficiency is proportional to kn^3). However, for queries to a data base system n is comparatively small, so that for choosing a parser the factor k becomes of major importance.

The Semantic Validity Test Should Be Performed Only After a Syntactic Analysis

The semantic validity of a query may be controlled in combination with the retrieval. Concurrent access to the data base

would have a far worse effect, since all dead ends during the analysis would add to the total retrieval time although not to the result. In spite of well known objections [51] quite a number of authors working at natural language access to data base systems [17,32] defend the principle of postponing the validity test. The number of syntactically correct but semantically meaningless constructions may be reduced by a special structure of the grammar (see next section).

The Subset of the German Language Defined

Based on the above demands a subset of the German language was defined for accessing the KAIFAS data base system by means of a context-free grammar. Following the works by Schott [20] a procedure for simplified morphemic analysis (without verbs) was developed. The parser was derived from the M. Kay-parser [52] since it best met the requirements of simplicity and speed (see next section, [38]). The translation from the natural language to the set algebraic language (MS in KAIFAS) follows traditional approaches. This process consists of lexical analysis, syntactical analysis, code generation, and transformations. Figure 1 illustrates the interaction between the steps. During each of these the following functions are performed:

- The query is divided into terminal symbols. When searching a dictionary of those, their corresponding representations on the set language level are found.

- The parser completes the syntactical analysis by means of the grammar.

- If the query is parsed to a sentence, the code generation will form an expression of the set language by using the terminal representations and the code fragments generated by the parser.

- Then transformations will be applied to this expression according to certain rules which will be explained in the next section.

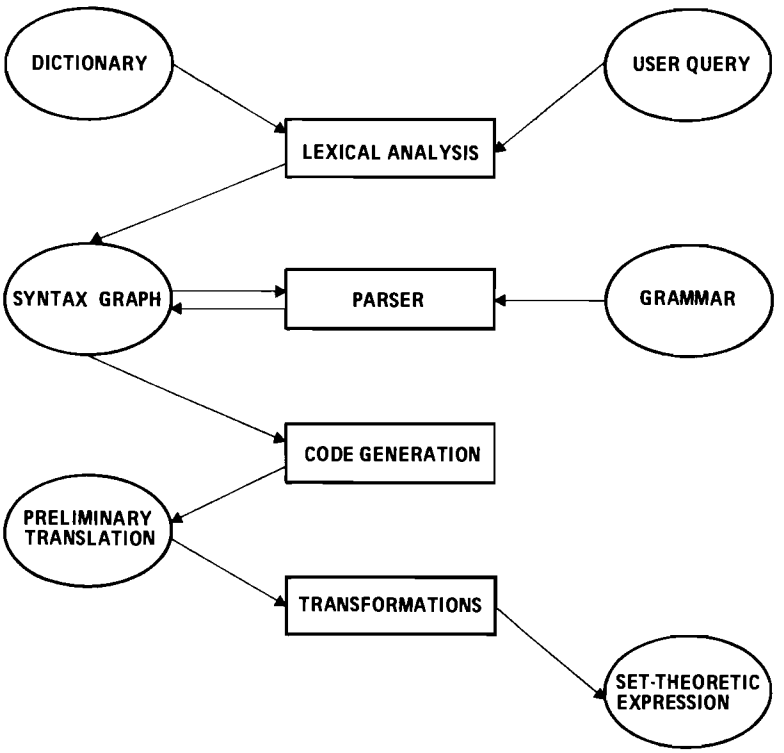


Figure 1. Translation to set language.

TRANSLATION OF NATURAL LANGUAGE INTO SET LANGUAGE

Type and Form of the German Grammar

Vocabulary

As pointed out earlier a context-free formalization of the natural language interface in KAIFAS was chosen. While a suitable subset of the German language can be described as a context-free language, it is just as important that this description be practicable, i.e. comprehensible and transparent to the designer and user, and easy to implement by the system. Practicability can only be achieved by using some additional tools.

Each context-free grammar contains a number of nonterminals in order to describe the syntactical phenomena of a language.

In many natural languages these tend to be quite extensive, e.g. the combinations of case, gender, and number. In order to limit the set of nonterminals in the grammar, so-called complex categories (based on REL [40,53]) are introduced. These may be considered schemas for nonterminals, and consist of a main category and a number of features. Traditionally [40,54,55], the main categories are related to syntactical phenomena such as noun or noun phrase, whereas features refer to secondary phenomena such as number or case. The values of a feature correspond to such phenomena, e.g.

number (1) = singular
case (2) = genitive .

Schemas denote sets of nonterminals; e.g.

$$N_{num,cas,gen}$$

denotes a set of 24 nonterminals all of them nouns. Complex categories may be partially ordered by assigning values to the features:

$$N_{num,cas(2,3),gen(1,2)}$$

is a more restricted schema than $N_{num,cas,gen}$, and denotes a set of only 8 nonterminals, since only two values are possible for each, cas(2,3) and gen(1,2). Assigning a single value to each feature of a complex category results in a single nonterminal.

The treatment of complex categories in KAIFAS is different from already existing approaches in several respects:

- Because of the restricted semantics in KAIFAS, main categories are chosen in accordance with semantic aspects (e.g. main category for sets ME, for relations RE). In this way we make sure that only semantically valid constructions are described by the productions of the grammar.

- A grammar whose main categories are based on semantical terms does not "naturally" reject sentences containing major errors such as missing congruence in number, gender, and case of adjectives and nouns, since these concepts are based on the traditional syntactical categories. The necessary syntactical aspects will be assigned to the features. As a result KAIFAS closely follows the correspondences

main categories - semantical aspects
features - syntactical aspects.

For practical reasons this classification cannot always be maintained. The set language, for example, contains (semantically) different types of quantifiers which in many productions are handled in the same (syntactical) way. The number of main categories and hence productions can be reduced by expressing the difference in type on the feature level so that only one main category will be defined for quantifiers.

- Only binary features are allowed:

  case features: *nom, gen, dat, acc*
  gender features: *mas, fem, neu*
  number features: *sin, plu*

  The binary values are designated by +/-. Then, operations with features may easily be expressed by logic formulas.

Table 3 provides a list of the main categories and features of the KAIFAS grammar. It is apparent from the figure that the main categories fall into two classes,

(a) object-categories and

(b) operator-categories,

corresponding to the classification of the set-theoretic language into objects and operators. Applying the same distinction to the terminals of the grammar results in

(a) object symbols which represent instances of the object types and

(b) operator symbols which represent the operators of the set-theoretic machine (the environment of the language in the sense of [56]).

Because the set of terminal symbols is both variable and large (approximately 50,000 objects in the pharmaceutical area) terminal productions are not made part of the grammar, but are maintained by means of a dictionary (lexical analysis is described later). The operator symbols form a fixed set, but some concordances can be identified, such as all operator symbols for quantifiers to which only one main category will be assigned (see above). The operator symbols are also included in the dictionary.

Table 3. Main categories and features.

Main categories

AF	AL - quantifier
DZ	measures
ED	term for evaluating measures
EI	measure units
IN	proper name
ME	set
MF	measure function
PR	operators: relations → sets (prepositions)
RE	relations
RP	DB-quantifier
RS	restriction of sets (relative clause)
SA	sentence
SF	EI/KE - quantifier
QU	other quantifiers
VO	relational operator
ZA	number

Features

Syntactical aspects:

mas	masculine
fem	feminine
neu	neuter
nom	nominative
gen	genitive
dat	dative
acc	accusative
sin plu	number
adj	adjective/noun
att	attributive
ajm	adjective-modified
pdt	predetermined (the drug)
prm	premodified (Peter's friend)
pom	postmodified (friend of Peter)
std	strong declination
svk	stops genitive concatenations

Semantical aspects:

qua	quantified
neg	negation-quantifiers (no: KE)
frw	interrogative (who, what)
mul div exp add	for arithmetic operators

Productions

The use of complex categories requires a similar extension of productions into complex productions:

$$NP_{cas,gen,num} \rightarrow Det_{cas,gen,num} N_{cas,gen,num} \quad .$$

This is a production schema from which one may derive a set of context-free productions when substituting suitable nonterminals for the complex categories. In doing so, the feature values have to meet certain conditions (e.g. congruence in case and number). Consequently, the complex rules are separated into a rewrite rule defined on main categories only, and a feature program specifying which combination of feature values may be assigned to the complex categories in this rule. The feature program consists of a test section specifying the conditions that the feature values of the complex categories in the right-hand part of the production have to meet in order to apply the rule, and an assignment section defining the feature values for the left-hand complex category. Test and specification could be done in list form [45] or by means of programs in a special programming language such as in KAIFAS.

In summary, a complex rule may be defined as follows:

(1) $\overline{V}_0 \rightarrow \overline{V}_1,\ldots,\overline{V}_p$	rewrite rule
(2) $A(V_1,\ldots,V_p)$	feature program (test)
(3) $Z(V_1,\ldots,V_p)$	feature program (assignment of the features of V_0)
(4) $S(V_1,\ldots,V_p)$	semantical part of the rule

$V_0,V_1,\ldots,V_p$ denote complex categories, $\overline{V}_0,\overline{V}_1,\ldots,\overline{V}_p$ their main categories. Table 4 provides a summary of the operators used in feature programs.

The semantic part is a term for defining the meaning of the complex rule. It consists of set-language symbols, and place-markers for the semantics of the complex categories in the rule. Since some semantical aspects are treated on the feature level, the semantic part may depend on conditions concerning features. These dependencies are defined by feature programs as well. Thus the semantic part $S(V_1,\ldots,V_p)$ of a complex rule may alternatively be phrased as

$$A(V_1,\ldots,V_p) \Rightarrow S'(V_1,\ldots,V_p) \qquad \text{(dependencies on features)} \quad ,$$

or

$$S'(V_1,\ldots,V_p) \qquad \text{(no dependencies)} \quad .$$

Table 5 gives an example of a complex rule.

A complex rule may be applied in the following steps (further details are provided later):

- matching the input string with the right-hand side of the rule;
- testing the right-hand features for acceptance; and
- if yielding "true", reduction to left-hand side and assignment of features and semantics.

Table 4. Operators in feature programs.

Test part:	
test	(⟨complex category⟩,⟨list of feature-values⟩) yields true, if the complex category has associated with it the feature-values specified, else false.
meq	(⟨complex category⟩,⟨complex category⟩,⟨list of features⟩) yields true, whenever at least one of the listed features agrees in both complex categories specified.
equ	(⟨complex category⟩,⟨complex category⟩,⟨list of features⟩) same as *meq*, but all features must agree.
∧,∨	logical connectives
Assignment part:	
(all assignments are to the complex category of the left rule part)	
zuw	(⟨list of feature-values⟩) assigns the feature-values specified.
cop	(⟨complex category⟩,⟨list of features⟩) copies the values of the features of the denoted complex symbols.
and	(⟨complex category⟩,⟨complex category⟩,⟨list of features⟩) assigns those feature-values that agree in both complex categories.

Table 5.

(1) ME → ME ME	rewrite rule
(2) *test* (ME^2,+*adj-att*) ∧ *test* (ME^3,-*adj*)∧ *meq* (ME^2,ME^3,*sin,plu*) *meq* (ME^2,ME^3,*nom,gen,dat,acc*)∧ *meq* (ME^2,ME^3,*mas,fem,neu*);	feature program (test)
(3) *zuw*(-*adj*), *and*(ME^2,ME^3,*sin,plu*), *and*(ME^2,ME^3,*mas,fem,neu*) *and* (ME^2,ME^3, *nom, gen, dat, acc*);	feature program (assignment)
(4) M∩ (ME^2,ME^3)	semantic part

The upper index (ME^2) serves for disambiguation of the complex categories of the production. The feature test excludes those adjectives and nouns that do not agree in number and in at least either gender or case. The resulting complex category is treated like a noun (-*adj*), most of the possible ambiguities in gender and noun are solved by the *and*- operator. The semantics of the production is the set-intersection.

Lexical Analysis

Assignment of Complex Categories

The dictionary contains all the object-symbols and all those operator-symbols that have been classified into types. Since the set of object-symbols is chosen in a user or application dependent fashion, these will not be defined until a user actually works with the system.

The lexical analysis fulfills two functions:

- assignment of a complex category to a terminal (assign a main category, assign feature-values),
- assignment of semantics (i.e. a terminal symbol of the set language).

A large number of syntactical ambiguities may be expressed within one complex category, such as the ambiguity arising by the German word "ein" (+*nom* or +*acc*):

ein $\xrightarrow{\textit{lexical analysis}}$ *QU-mas-fem+neu+nom-gen-dat+acc+sin-plu* ,

where the list of feature-values may be considered a conjunctive logical expression. If disjunction is needed as well, a conjunctive normal form is used. An example from the German language is:

$$drageeförmigen \rightarrow \begin{cases} ME+mas-nom+gen+dat+acc+sin-plu \\ ME+fem+neu-nom+gen+dat-acc+sin-plu \end{cases} .$$

Here the accusative case is allowed for masculine, but not for neuter or feminine.

Morphology

The multitude of inflections in the German language does not allow for storing in a dictionary all word forms to be derived from a large user-vocabulary. Rather, the dictionary only contains the word stems. Reduction from inflective form to word stem is done by algorithmic means (morphological analysis). The exclusion of verbs simplifies the problem. A word stem is defined as follows:

- nouns: nominative-singular form
- adjectives: attributive form.

Also, the morphological analysis must determine the syntactical structure of a terminal (gender, case, etc.). Different approaches have been published for solving this problem for the German language [20], but all of them require that extensive linguistic information be supplied with each word in the dictionary which can hardly be expected from a casual user. Therefore, when defining a word stem in KAIFAS the user will only be required to specify a minumum of information, namely:

- object-class of a word
- gender
- noun/adjective
- singular and plural forms of the word.

The word may then be assigned to a specific morphemic class (see [20]). This class contains all morphemic endings that may be attached to the word stem. Each morphemic ending will determine one or more syntactical structures (set of feature-values) for all these terminals that contain this ending. By explicitly storing the plural forms of terminals the highly problematical reduction of terminals involving mutation of vowels becomes unnecessary. The additional storage space required may be tolerated, since plural forms occur in case of set- and relation-identifiers only. Table 6 presents an example of a morphemic class. In order to save storage space in the dictionary, the syntactical structure of a word stem will also be defined by morphological analysis. Thus any morphemic class will contain an entry for the null ending "ε".

Table 6. Morphemic class for articles singular (e.g. "kein").

Ending	Syntactical structure
ε	+mas-fem-num+nom-gen-dat-acc -mas-fem+neu+nom-gen-dat+acc
e	-mas+fem-neu+nom-gen-dat+acc
es	+mas-fem+neu-nom+gen-dat-acc
em	+mas-fem+neu-nom-gen+dat-acc
en	+mas-fem-neu-nom-gen-dat+acc
er	-mas+fem-neu-nom+gen+dat-acc

In Table 7 the complete lexical analysis of a query is illustrated. The syntactical structure of a terminal can be highly ambiguous due to the lexical analysis. The feature programs, however, allow for easy disambiguation as demonstrated by the example of Table 8.

Table 7. Result of lexical analysis for an entire sentence.

Word	Main category	Features	Set-theoretic representation
Welche	QU	+mas+fem+neu+nom+acc+plu	DB
	QU	+fem+nom+acc+sin	DB
dragee-	ME	+mas+neu+gen+dat+acc+sin	M25
förmigen	ME	+fem+neu+gen+dat+sin	M25
	ME	+mas+fem+neu+nom+gen+dat+acc+plu	M25
Psycho- pharmaka	ME	+neu+nom+gen+acc+plu	M4
haben	⟨terminal⟩	-	
Depression	IN	+nom+dat+acc	I64
als	⟨terminal⟩	-	Ng
Indikation	RE	+fem+nom+gen+dat+acc+sin	R8

Table 8. Disambiguation.

Rule: $ME_1 \rightarrow ME_2 \quad ME_3$

Meq(mas,fem,neu,ME_2,ME_3)∧ Meq(nom,gen,dat,acc,ME_2,ME_3)∧

Meq(sin,plu,ME_2,ME_3);

And(mas,fem,neu,ME_2,ME_3),And(nom,gen,dat,acc,ME_2,ME_3),

And(sin,plu,ME_2,ME_3);

M∩(2,3);

applied to

(1) ME +mas-fem-neu-nom+gen (drageeförmigen)
+dat+acc+sin-plu

(2) ME -mas+fem+neu-nom+gen "
+dat-acc+sin-plu

(3) ME +mas+fem+neu+nom+gen "
+dat+acc-sin+plu

(4) ME -mas-fem+neu+nom+gen (Psychopharmaka)
-dat+acc-sin+plu

Because of number (sin,plu) the feature test accepts combinations (3) and (4) only. The feature-assignment yields:

(5) ME -mas-fem+neu+nom+gen (drageeförmigen Psychopharmaka)
-dat+acc-sin+plu

Rule: $ME_1 \rightarrow QU_2 \quad ME_3$

Meq(mas,fem,neu,QU_2,ME_3)∧ Meq(nom,gen,dat,acc,QU_2,ME_3)∧

Meq(sin,plu,QU_2,ME_3);

And(mas,fem,neu,QU_2,ME_3), And(nom,gen,dat,acc,QU_2,ME_3),

And(sin,plu,QU_2,ME_3):

2(x,3,#)

applied to

(6) QU +mas+fem+neu+nom-gen-dat (welche)
+acc-sin+plu

(7) QU -mas-fem-neu+nom-gen-dat "
+acc+sin-plu

(5) ME -mas-fem+neu+nom+gen (drageeförmigen Psychopharmaka)
-dat+acc-sin+plu

Only combination (6)/(5) is accepted.

Result: ME -mas-fem+neu+nom-gen (welche drageeförmigen Psychopharmaka)
-dat+acc-sin+plu

Thus all ambiguities with the exception of case (nominative/accusative) are resolved.

Lexical Analysis Algorithm

Lexical analysis for a word $x = x_1,\ldots,x_k$ is carried out according to the simple algorithm outlined below:

For $l = 0,1,2,\ldots,\min(k-1,3)$:

$x' = x/x_{k-l},\ldots,x_k$ (delete $x_{k-l},\ldots,x_k$ from x) .

If x' is found in the dictionary and $x_{k-l},\ldots,x_k$ belongs to the morphemic class of x', then assign to x':

(1) the main category of the dictionary entry,

(2) the features of the dictionary entry,

(3) the features defined by the entry of $x_{k-l},\ldots,x_k$ in the morphemic class,

(4) the semantics specified in the dictionary.

This algorithm is applied to each terminal of a query. The result will be converted to a form suitable for the ensuing parsing process.

Parser

The parser completes the syntactical and semantic analysis of a query. According to what has been said so far it has to meet the following conditions:

- The parser has to recognize context-free languages.
- It must be able to operate on complex categories and rules.
- The storage space and execution time required for the analysis should be kept small in comparison to the requirements of the entire retrieval process.
- Furthermore a syntax-directed approach is needed for parsing that is independent of a special grammar. This is due to the fact that constructing a grammar for natural languages in an approximative process. The grammar will be continuously modified and enlarged in order to eliminate wrong constructions or to extend the set of permissible sentences.

Several parsers are known to meet the first and last conditions, whereas an adaptation to the second is always necessary. Among these, Earley's parser for context-free grammars suggests itself [48]. However, adapting an improved version of this parser to complex categories and rules resulted in an unwieldy algorithm violating the third condition (see [57]).

Therefore a parser based on the ideas of Kay [52] was developed. The original algorithm is capable of operating on general rewrite rules but was restricted to context-free grammars expressed in our complex notation. Only a short introduction to this parser will be presented, for details we refer you to [38].

Figure 2 represents a typical parsing graph as generated by the parser. The graph contains n+1 vertices for a query consisting of n words. Every edge of the graph is labeled by a complex category and its semantics.

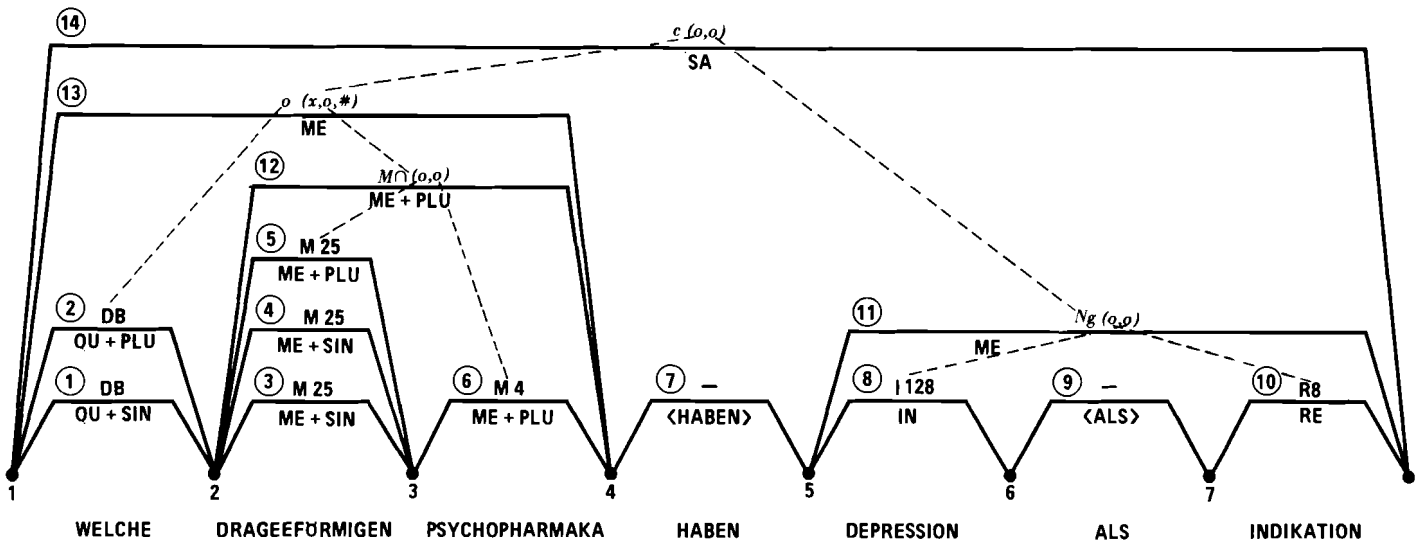
During lexical analysis an initial parsing graph is constructed (heavy lines in Figure 2). It contains edges only between vertices k and k+1 ($1 \leq k \leq n$). The number of edges between two vertices is l, where l is the number of complex categories assigned to the k-th terminal in a query.

The parser operates on the initial parsing graph as follows: Starting at vertex k, for all sequences of edges from k to vertices k' ($k < k' \leq n+1$) the parser compares the main categories within the labels with the right-hand sides of all complex rules. On total agreement with a rule r, the parser performs the following steps:

- The feature program of rule r operates on the complex categories in the sequence of edges.
- If the test yields "true", the parser produces a new edge between the starting and ending vertices of the sequence of edges. The new edge is labeled by the left-hand side of the rewrite rule and by the features obtained from the assignment section in the feature program of r.
- The edge is additionally labeled by the semantics of the rule with all place-markers replaced by pointers to the semantics of the complex categories in the sequence of edges.

This process is repeated for all vertices from right to left down to vertex 1.

The parsing of a query will prove successful if there is an edge between vertices 1 and n+1 labeled by the axiom of the grammar (in this case SA).



Grammar rules applied (the rewrite rule and semantics only):

ME → IN als RE:
Ng(RE,IN)

ME → ME_2 ME_3:
M∩(ME_2,ME_3)

ME_1 → OU ME_2:
Ou(x,ME_2,#)

SA → ME_1 haben ME_2:
c(ME_1,ME_2)

Only those rules are stated whose application does not result in dead-ends.

Figure 2. Parsing-graph. Only those features are listed that prevent a rule from being applied.

In Table 9 a special numbering (①,②) shows the order by which the edges have been generated. The exact structure of the parse can be derived by means of the pointers that connect the semantics. Obviously, the pointer structure saves storage space over a solution generating complete code fragments for each edge.

Table 9. Transformations.

Welche drageeförmigen Psychopharmaka haben Depression als Indikation?
Preliminary translation: $\subset(DB(x,M\cap(M25,M4),\#),Ng(R8,I128))$
Transformation: Quantifier DB is placed in front of the expression, $\subset$ transformed to $\in$.
$DB(x,M\cap(M25,M4),\in(x,Ng(R8,I128)))$
Welche Indikationen welcher Medikamente sind Psychosen?
Preliminary translation: $\in(DB(x_1,Vg(R8,DB(x_2,M29,\#)),\#),M30)$
Transformation: Both quantifiers are placed in front but in reverse order.
$DB(x_2,M29,DB(x_1,Vg(R8,x_2),\in(x_1,M30)))$

Code Generation

On code generation the fragments are assembled into one or more expressions of the set-theoretic language depending on the ambiguity of the query. These expressions form the result of query translation.

Transformations

Applying the code generation process to the parsing graph in Figure 2 results in the following expression:

$$\subset(DB(x,M\ (M25,M4),\#),Ng(R8,I128))\quad .$$

This expression is not well formed, since the quantifier is not the left-most operator (prenex normal form). # serves as a place-marker for the scope.

Problems of this kind and other syntactical properties of the set language pose difficulties when handled by the translation mechanism introduced above [38]. Other examples of this nature are the following:

- Nesting of quantifiers in the set language is subject to certain rules that control their relative position within an expression: set quantifiers like DB must appear in front of logical quantifiers like AL, EI.

- Difficulties arise from the difference in relative position of operator symbols for quantifiers within natural German and in that of their corresponding quantifiers within the set-theoretic equivalent:

  Which remedies for which diseases are prescription drugs?

  $$DB(x_1, M_{\text{diseases}}, DB(x_2, Vg(R_{\text{remedy}}, x_1), \in(x_2, M_{\text{prescription drug}}))) \ .$$

These problems can be solved by means of grammar rules, but then the grammar proves impractical (see [38]). Thus these problems are deferred to an analysis phase that takes place after completion of the parsing process and hence after application of the grammar rules.

A solution could be based on the tree-like pointer structure of the semantic fragments, which we shall call a semantic tree. The semantic tree has then to be transformed by suitable rules such that the linear expression derived by code-generation puts the quantifiers in the right order. Transformations of parsing-trees are usually formulated by means of transformational grammars, but implementing these requires large efforts in time and personnel [58].

Moreover, the problems just discussed form a trivial subset of the general transformation problem. It can be shown that for every transformation rule defined on semantic trees and needed here, there exists a corresponding rule defined on the linear form, the expression. In place of the traditional approach where basic trees are given as arguments, these rules contain an expression pattern that must be contained in the expression to be transformed. For example, the pattern

$$DB_1(x_1, Vg(R, DB_2(x_2, M, \#)))$$

is transformed into:

$$DB_2(x_2, M, DB_1(x_1, Vg(R, x_2), \#)) \ .$$

Manipulations of linear expressions are easier to do than transformation of trees. Thus the transformations are postponed until after code-generation. The transformations may be formulated by means of a string-manipulation language forming a set of procedures. (Our work in this direction is discussed in [59].) These procedures are integrated in the system and executed after code generation. Table 9 shows some examples of transformations.

CONCLUSIONS

The linguistic techniques discussed in this paper have been implemented at the University of Karlsruhe on a Burroughs 6700 as part of the KAIFAS information system. Some experience in their usefulness has been gained by applying the system to a pharmaceutical data base containing data on a part of the drugs available on the German pharmaceutical market (about 8000 [60]). This data base was applied by experts inexperienced in data processing via the natural German interface.

One of the purposes of this implementation was to test the premises defined earlier for their validity. A context-free definition of the natural language interface proved sufficient for this application. Whether this is true in general can only be decided if one included verbs in the interface. The descriptive power of the context-free grammar in the system was even unnecessarily large, because the users when working with the system tended to use several short queries successively instead of a single long one, i.e. they solved their problems in steps. For this reason relative clauses could eventually be excluded from the query language. Instead, the possibility of references to queries stated before is needed involving solutions to the well known problem of pronouns.

The morphological analysis proved to be sufficient, too. All correct inflectional forms were detected and reduced. The simple approach will not guarantee, however, that a syntactical incorrect inflectional form will be refused under any circumstances.

The M. Kay parser, which we restricted to context-free languages, turned out to be a very simple algorithm. One can show that the algorithm is superior to Earley's parser with respect to processing time for short sentences in the neighborhood of ten words or less. Consequently, the M. Kay parser is particularly suited to the stepwise user approach mentioned above.

REFERENCES

[1] Winograd, T., *Five Lectures on Artificial Intelligence*, Computer Science Department, Stanford University, 1974.

[2] Smith, L.C., Artificial Intelligence in Information Retrieval Systems, *Information Processing & Management*, 12 (1976), 189-222.

[3] Quillian, R., Semantic Memory, in M. Minsky, ed., *Semantic Information Processing*, MIT Press, Cambridge, Mass., 1968.

[4] Schank, R., Identification of Conceptualizations Underlying Natural Language, in R.C. Schank and K.M. Colby, eds., *Computer Models of Thought and Language*, Freeman, San Francisco, 1973.

[5] Charniak, E., *Toward a Model of Children's Story Comprehension*, MIT Artificial Intelligence Laboratory, Cambridge, Mass., 1972.

[6] Cherniavsky, V., *On Algorithmic Natural Language Analysis and Understanding*, presented at the Advanced Course on Data Base Languages and Natural Language Processing, Freudenstadt, September 1976.

[7] *Proceedings of the IFIP-TC-2 Working Conference on: Modelling in Data Base Management Systems*, Freudenstadt, January 1976.

[8] Durchholz, R., and G. Richter, Concepts for Data Base Management Systems, in *Proceedings of the IFIP-TC-2 Working Conference on "Data Base Management Systems"*, North-Holland, Amsterdam, 1974.

[9] Codd, E.F., A Relational Model of Data for Large Shared Data Banks, *Commun. ACM*, 13 (1970), 377-387.

[10] Taylor, R.W., and R.L. Frank, CODASYL Data Base Management Systems, *ACM Computing Surveys*, 8, 1 (1976), 67-104.

[11] CODASYL-DBTG, *Data Base Task Group Report*, New York, 1971.

[12] Abrial, J.R., Data Semantics, in *Proceedings of the IFIP-TC-2 Working Conference on "Data Base Management Systems"*, North-Holland, Amsterdam, 1974.

[13] Date, C.J., *An Introduction to Database Systems*, Addison-Wesley, Reading, Mass., 1975.

[14] Montgomery, C.A., Is Natural Language an Unnatural Query Language?, in *Proceedings of an ACM National Conference*, Association of Computing Machinery, New York, New York, 1972.

[15] *Proceedings of the Advanced Course on Data Base Languages and Natural Language Processing*, Freudenstadt, September 1976.

[16] Krägeloh, K.D., and P.C. Lockemann, Hierarchies of Data Base Languages: An Example, *Information Systems*, 1 (1975), 79-90.

[17] Codd, E.F., Seven Steps to Rendezvous with the Casual User, *Proceedings of the IFIP-TC-2 Working Conference on "Data Base Management Systems"*, North-Holland, Amsterdam, 1974.

[18] Malhotra, A., *Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis*, MIT Project MAC, Cambridge, Mass., 1975.

[19] *Data Base*, 8, 2 (1968).

[20] Schott, G., *Automatische Analyse der Flexionsmorpheme deutscher Substantive*, Bericht Nr. 7210, Technische Universität München, Abteilung Mathematik, Gruppe Informatik, 1972.

[21] *Zur maschinellen Syntaxanalyse*, Forschungsberichte, Institut für deutsche Sprache, Mannheim, Band 18.1, 18.2, 19., Narr-Verlag, Tübingen, 1974.

[22] Salton, G., *Automatic Information Organization and Retrieval*, McGraw-Hill, New York, 1968.

[23] Josselson, H.L., Automatic Translation of Languages Since 1960: A Linguist's View, *Advances in Computers*, 11 (1971), 1-58.

[24] Turing, A.M., Computing Machinery and Intelligence, *Mind*, 59 (1959), 433-460.

[25] Fodor, J.A., and J.J. Katz, The Structure of a Semantic Theory, in J.A. Fodor and J.A. Katz, eds., *The Structure of Language*, Prentice Hall, Englewood Cliffs, New Jersey, 1964.

[26] Bobrow, D.G., A Question-Answering System for High School Algebra Word Problems, *Procs. AFIPS Fall Joint Comp. Conf.*, 26 (1964), 591-614.

[27] Raphael, B., SIR, *Procs. AFIPS Fall Joint Comp. Conf.*, 26 (1964), 577-589.

[28] Green, B.F., et al., BASEBALL, in F.A. Feigenbaum and J. Feldman, eds., *Computer and Thought*, McGraw-Hill, New York, 1963.

[29] Weizenbaum, J., ELIZA, *Commun. ACM*, 9 (1966), 36-45.

[30] Coles, L., and L. Stephen, An On-Line Question Answering System with Natural Language and Pictorial Input, in *Procs. ACM 23rd National Conference*, Association of Computing Machinery, New York, New York, 1968.

[31] Winograd, T., *Understanding Natural Language*, Academic Press Inc., New York, 1972.

[32] Woods, W.A., Procedural Semantics for a Question Answering Machine, *Proc. AFIPS Fall Joint Comp. Conf.*, 33 (1968), 457-471.

[33] Woods, W.A., Progress in Natural Language Understanding - An Application to Lunar Geology, in *Proc. National Comp. Conf.*, 1973.

[34] Mylopoulos, J., S. Schuster, and D. Tsichritzis, A Multi-Level Relational System, in *Proc. National Comp. Conf.*, 1975.

[35] Astrahan, M.M., et al., System R: Relational Approach to Database Management, *ACM Transactions in Database Systems*, 1, 2 (1976).

[36] IMS 2, in *Kurzbeschreibung von Information Storage and Retrieval Systemen*, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1973.

[37] Todd, S., *Integrated Architecture for Transaction Specification and Optimization in Relational Data Base Systems*, presented at the Summer School on Data Base Technology, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1976.

[38] Krägeloh, K.D., *A Multi-Level System Architecture with Natural Language Interface*, Ph.D. Thesis, University of Karlsruhe, 1976 (in German).

[39] Kellog, C.H., A Natural Language Compiler for Online Data Management, *AFIPS Fall Joint Comp. Conf.*, 33 (1968), 473-493.

[40] Thompson, F.B., P.C. Lockemann, B. Dostert, and R.S. Deverill, REL: A Rapidly Extensible Language System, in *Proceedings of the 24th National ACM Conference*, Association of Computing Machinery, New York, New York, 1969.

[41] Bachman, C.W., The Programmer as Navigator, *Commun. ACM*, 16, 11 (1973), 653-658.

[42] Lacroix, M., and A. Pirotte, *ILL: An English Structured Query Language for Relational Data Bases*, M.B.L.E. Research Laboratory Report, Brussels, 1976.

[43] Feldman, J.A., and P.P. Rovner, An ALGOL-Based Associative Language, *Commun. ACM*, 12, 8 (1969), 439-449.

[44] Goos, G., *Programmkonstruktion*, internal report, University of Karlsruhe, 1974.

[45] Kratzer, A., E. Pause, and A. v. Stechow, *Einfürhung in die Theorie und Anwendung der generativen Syntax*, Athenaeum Verlag, Frankfurt, 1974.

[46] *PASSAT*, Systembeschreibung Siemens PBS4004, Munich, 1973.

[47] Steinacker, I., *Dokumentationssysteme*, De Gruyter, Berlin-New York, 1975.

[48] Earley, J.C., *An Efficient Context-Free Parsing Algorithm*, Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, Penn., 1968.

[49] Kasami, T., An Efficient Recognition and Syntax Analysis Algorithm for Context-Free Languages, University of Illinois, Urbana, 1966.

[50] Younger, D.H., Recognition and Parsing of Context-Free Languages in Time n, *Information and Control*, 10 (1967), 189-208.

[51] Simmons, R.F., Natural Language Question Answering Systems: 1969, *Commun. ACM*, 13, 1 (1970), 15-30.

[52] Kay, M. Experiments with a Powerful Parser, presented at Deuxième Conference sur le Traitement automatique des langues, Grenoble, 1967.

[53] Dostert, B.H., and F.B. Thompson, How Features Resolve Syntactic Ambiguity, in *Proceedings of the Symposium on Information Storage and Retrieval*, University of Maryland, College Park, 1971.

[54] Brockhaus, K., *Automatische Übersetzung*, Vieweg Verlag, Braunschweig, 1971.

[55] Wulz, H., *ISLIB - Ein Informationssystem auf linguistischer Basis*, internal report, Institut für deutsche Sprache, Abteilung linguistische Datenverarbeitung, Mannheim, 1975.

[56] Thompson, F.B., English for the Computer, *Procs. AFIPS Fall Joint Comp. Conf.* (1966), 349-356.

[57] Wohlleber, W., *Ein Parser für die Analyse natürlicher Sprache*, Diplomarbeit, University of Karlsruhe, 1973.

[58] Friedman, J., *A Computer Model of Transformational Grammar*, American Elsevier Publishing Company Inc., New York, 1971.

[59] Mathis, C., *Entwurf und Implementierung einer textverarbeitenden Sprache*, Diplomarbeit, University of Karlsruhe, 1975.

[60] Bundesverband der pharmazeutischen Industrie, Frankfurt, *ROTE LISTE 1975*, EDITIO CANTOR, Aulendorf/Württ., 1975.

Catering for the Experienced and the Naive User

M. King, P. Dell'Orco, and V.N. Spadavecchia

1. INTRODUCTION

The system described here has been developed on a theoretical base that many may perhaps find uncomfortable. We have chosen to develop and implement the data base model (and the formal query language that interrogates the data base) independently of any consideration of the natural language front end that will ultimately be attached to the system. There are, or so it seems to us, good reasons for doing this. First the data base model we use is well known and theoretically well developed, so that its implementation is, whilst not easy, at least relatively straightforward. Secondly the system is designed with two sorts of user in mind: the experienced computer user who is prepared to accept a degree of formalization in the query language he must use and the more naive user for whom any insistence on strict format or (to him) counterintuitive modes of question formulation constitutes a serious difficulty. These two users determine two types of query language: the formal query language designed for the experienced user, a wide subset of natural language designed for the naive.

Although the formal query language is an obvious bridge between the naive user and the system, it would clearly be wasteful to design one entire system for the experienced user and a second, different, entire system for the nonexperienced. So we have regarded the natural language analysis as basically a type of encoding, transforming natural language queries into formal language queries. Once that decision is made it becomes even more obvious that, in terms of research strategy, it is better to implement the formal query language first, since it is shared by both users. It is also true that there is a strong sense in which the data base and its associated formal query language are logically prior to the natural language. The type and structure of the information contained in the data base determine in large part the subset of natural language to be dealt with, the formal query language determines the output structures that are to be produced by the natural language analyzer.

The present state of development of the system is summarized in Table 1. The system has been designed to be quite modular. Hence Table 1 is fairly accurate in that what appear to be separate modules connected by narrow interfaces are precisely that. In particular natural language analysis has been divided into two sections.

Table 1.

Input	Module	Output	State of Development
Natural language queries	Natural language analyzer	Intermediate semantic representation	Designed, not yet programmed
Intermediate semantic representation	Translator	Formal language queries	Being designed
Formal language queries	Interpreter, data base interrogation	Answers	Implemented

This paper gives an overview of the whole system using the formal query language as a pivot for the description, since it so clearly forms an interface between the natural language analysis part of the system and the formal part.

Section 2 describes the data base model itself. Section 3 argues briefly the case for allowing natural language input and gives a sketchy view of the general characteristics of the natural language analyzer. The main thrust of the paper is contained in Sections 4 and 5. Section 4 describes the formal query language, although not in full detail. Section 5 takes the intermediate semantic representation as established by the natural language analyzer and shows how it is transformed into the formal query language representation. One caveat should be entered here. The present translation algorithm is still crude. We anticipate that much further work will be required before it can be regarded as adequate. Nonetheless we believe that the current version is a reasonable first approximation and that its general outlines will remain substantially unchanged. Some more detailed criticism of the translation algorithm comes in the conclusion, which also outlines future development plans.

2. THE DATA BASE

The data base model used in this system is a relational model, based on the work of Codd [1]. This model was chosen because it is well developed theoretically, semantically complete, and logically transparent. Thus a user who wishes to deal directly with the data base finds it easy to conceptualize the structure of the base and to see how interconnections between different sets of information can be realized.

As indicated by the name, a relational data base is a base organized into a set of relations, each of which consists of a set of domains. The easiest way to visualize this sort of organization is to imagine each relation as a table, with the domains

specifying attributes of the entity represented by the relation and forming, as it were, column headings (Table 2). A row in such a table represents a particular instance of the entity described by that relation, and the values in the "boxes" defined by column and row indicator give the specific values of the attributes represented by the domains for this particular instance. All this sounds rather complicated. That it is really quite simple can be seen from Table 2, where the relation EMP (employee) of an example data base is shown.

Table 2. Relation EMP.

	CODE	NAME	DEPT	MANAGER	SALARY	COMMISSION
1	376	JONES	TOY	5247	3000	2
2	4923	MONTE	PERSONNEL	632	4050	
3	589	SCHMIDT	FASHION	219	2560	2.5
4	7235	ROBUL	HARDWARE	1523	3500	2

Here the domains are CODE (employee's code number within the organization), NAME (employee's surname), DEPT (the department in which he works), MANAGER (the code of his manager), SALARY (which needs no gloss), and COMMISSION (the percentage of his sales he takes in commission). Thus the employee represented by row three is called Schmidt, has code number 589, works in the fashion department, has as his manager another employee with code 219, earns 2560 a month and gets 2.5% commission on his sales.

Relations are not isolated. Often the values of a domain in one relation map onto the values of another domain in another relation, so that there is a logical path between the two relations. Clearly this can be extended to more than two relations. Part of the implementation of a relational data base involves providing facilities whereby an optimal path may be found when necessary. In general the user of the system need not know about connectivities between relations. When necessary the inference making part of the formal query language analyzer will detect the need to provide a path between relations, will determine what the path is, and will fill it in.

Throughout this paper we shall constantly refer to the mini data base given as an example in Table 3. It is part of a larger data base modeling a department store. The formal query language interrogates the data base. We return to that in Section 4.

Table 3.

Relation Name	Domains
EMP	CODE, NAME, DEPT, MANAGER, SALARY, COMMISSION
SALES	DEPT, ITEM, VOLUME, COST
SUPPLY	SUPPLIER, ITEM, VOLUME
DEPARTMENT	DEPTNO, PRODUCT, TELNO, LOC

3. THE NATURAL LANGUAGE ANALYZER

Given the notoriously intractable problem of natural language analysis, an obvious first question is why we should want to allow natural language input at all, especially since considerable effort has gone into designing and implementing a useful and comfortable formal query language for this same system. This is not the place to engage in lengthy polemic, but, briefly restated, the chief and most compelling argument lies in the fact that a user who is not a computer specialist finds input in a formal language, however well designed, sufficiently repugnant to discourage him from using the system. The amount of effort required to develop a large data base system is only worthwhile if the resulting system can be used by a wide variety of nonspecialist users, who must therefore be specially catered for by the provision of facilities that will accept as wide a range of natural language input as possible.

Although a decision to allow natural language input shows commendable friendliness to prospective users, it makes the system designer's task a great deal more difficult. The most critical decision he must make concerns the choice of an appropriate question analysis algorithm. No existing computer system deals with natural language in its full generality, but systems based on semantic methods of analysis seem to be considerably more powerful than syntactically based systems, which are subject to a number of severe drawbacks. First there is the constant danger of combinatorial explosion in any (otherwise blind) syntactic parser general enough to deal with an adequately large subset of natural language. Several ways have been suggested to overcome this problem (see, for example, [2]), but their success is still in doubt. The further problems of word sense and structural ambiguity and of determining anaphoric reference are even in principle insoluble for a purely syntactic parser. To see this, consider the sentence

> Give me the name of any employee who works in the personnel department whose salary is more than 3000 francs a month.

It is quite obvious to any person reading this that the "whose" refers to the employee, and not to the personnel department. Yet there is no syntactic rule that could be used to determine this. It depends on the semantic fact that employees earn salaries, departments do not. Indeed there could be no syntactic rule, since, with a different context, the reference can just as easily go the other way:

> Give me the name of any employee who works in the personnel department whose head office is in Rome.

Although this sentence is, in fact, genuinely ambiguous, it tends to be interpreted with the "whose" referring to the department.

A further argument comes from the inability of syntactic parsers to tolerate input that is even mildly ungrammatical (grammaticality being defined, of course, by the system itself). A very simple typing mistake, such as typing "whom" for "who" will normally be enough to break a syntactically based system. Using semantics as a basic tool in the analysis allows a greater tolerance of imperfect input.

In most essentials the analysis method used by this system is an adaptation of Wilks's [3] preference semantics system. This was originally developed as a framework for translation between two natural languages, English and French. Since its structure was extremely modular, by breaking it down into two main sections, the first of which dealt with the analysis of the input text and the second with the generation of the French translation from the intermediate semantic representation established by the analysis routines, it has been possible for us to take the first of these two main modules and use it as a basis for our own analysis. The intermediate representation thus established forms the input to a phase that translates it into the formal query language. The translation phase best reveals the major modifications made by us to the original Wilks's system and is therefore dealt with at some length in Section 5.

It is assumed here that the reader has some aquaintance with Wilks's system. Its general outline is recapitulated only to refresh the memory, and no pretence is made that an adequate account is given. The reader in search of more detail can find a very detailed description in [4]; [3] is briefer and more easily readable.

The crux of the whole system is the notion of semantic preference. Individual word senses are represented by semantic formulae that are structured organizations of semantic primitives. Within a formula a particular word sense may express a preference: a verb, for example, may express a preference for an animate subject, or an adjective a preference for being a quality of a

physical object. Such preferences are extensively used in deciding on a reading for a text, both at the level of determining overall structure and at the level of disambiguation or of reference determination. It is important however to remember that preferences are *only* preferences and not stringent semantic restrictions. If a preference cannot be satisfied, the system does not automatically reject the text as nonsensical but attempts an analysis based on fulfilling the maximum number of preferences possible. If this were not so, the system would break down when confronted by perfectly normal and comprehensible sentences that involve a word used in any but its most standard (arbitrarily defined to be standard) sense.

Given an understanding of this basic principle, we can, for the purposes of this paper, skate over the higher-order analysis of the input text. An input sentence is first broken down into fragments, each of which corresponds intuitively to a basic message or unit of information. Possible basic messages are represented within the system by triples of semantic primitives, and are called templates. MAN BE KIND, for example, is the basic message corresponding to any phrase of the general form "a person is an x"--like "Socrates is mortal" or "Children are normally happy". The templates are linked by case ties into higher order structures covering the whole of the input text.

Instead of trying to explain this in detail, let us consider an example. The sentence is one relevant to the example data base given in Section 2. We shall use the same sentence later to illustrate the translation algorithm.

> "Give me the name of the employee working in the department whose produce is XYZ and whose salary is 5000."

By the time the intermediate semantic representation is formed the reference of the two occurrences of "whose" has been resolved, so that the representation given diagrammatically and without semantic formulae is as shown in Figure 1.

Some commentary on this is clearly necessary. The text has been broken down into fragments, each of which is represented by a single template. Fragments of the text which do *not* form a complete template but are dependent on other text elements filling main positions in a template are attached to the elements on which they are dependent, and the nature of the dependency is given. Thus "me", the indirect object of "give" in the first template is marked as being in the RECIpient case relationship to "give". Similarly genitive constructions are marked by the possessive case (templates 1, 4, and 5). Relationships between templates are also specified by case-markers. APPOsitive is the case that tells us that the template attached expresses a restriction on the general class denoted by the element to which the tie is

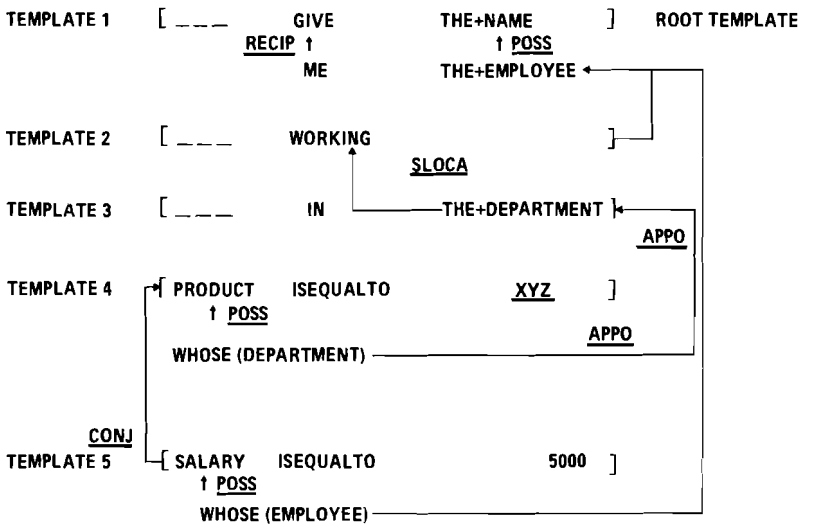


Figure 1.

made. SLOCAtive expresses the space-location case--the activity denoted by the element to which the tie is made takes place in the location given in the tied template. CONJunction is a syntactic marker rather than a case proper and is used in the obvious intuitive situations.

We shall return to this example sentence in Section 5.

4. THE FORMAL QUERY LANGUAGE

In this section we shall mainly be concerned with the formal query language (AQL) used to interrogate the data base. It is worth saying a little first about the other components of the system not mentioned otherwise. A component called the relational memory system (RMS) is used to map the conceptual, external view of the data base into internal machine terms. The RMS organizes the memory space used by the data base and provides a fast algorithm for efficient storage and retrieval of items. An interpreter, written in APL, interprets the formal query language. The basic idea inderlying its design is to allow the user to write a descriptive statement that is then translated by the interpreter into a sequence of routine calls which interface with the RMS. The interpreter also allows for default options in the formulation of a query. Obvious actions, like the quoting of constants or connecting any domain unique to a particular relation to that relation, are carried out by the system instead of having to be specified by the user. Similarly the interpreter allows a user to refer to a given domain by means of a synonym

or by a definition once such synonyms or definitions have been declared to the system. When a query is genuinely ambiguous or is incompletely specified the user is offered a "menu" and is asked by means of it to supply the missing information.

Two further features make AQL an even easier language to use: An inference maker, incorporated in the interpreter, avoids the user having to specify how navigation between relations is to be performed (mentioned in Section 2). The language is also nonprocedural in the sense that the user needs to specify only what has to be retrieved rather than how to get it.

More detailed description of AQL is best done by examples, but it should be noted that in a limited space it is not really possible to give an adequate idea of the power of a complex formal language. Throughout the examples the mini data base given in Section 2 is assumed to be the data base available.

A first, very simple example will illustrate some basic features of AQL.

Q1. Find the salaries of the employees working in department 139.

```
Q
(SALARY OF EMP)
WHEN
DEPT EQ 139
```

The syntax of such a query is very like that of an APL statement. The keyword Q denotes a query. The part of the query to the left of WHEN is the request list. It consists of a list of one or more domain names (linked by WITH when there is more than one) referred to the proper relation by the function OF. To the right of WHEN is the condition list, consisting of one or more elementary conditions separated by the logical functions AND, OR, NOT. An elementary condition is simply a domain name followed by a comparison operator followed by a value or a list of values. If the user wants all the domains of a particular relation he may write ALL INFO instead of the list of requested domains. If he wants all the information contained in a domain he may replace the list of conditions by the function ALL. When only the request list is specified WHEN ALL is assumed by default. ALL assumes its argument to be the relation name last mentioned before the occurrence of ALL.

The result of Q1 is a single column matrix that will be given the name SALARY, and will become the value of an APL variable of the same name in the workspace. This is always true: the result of a query that directly manipulates the data base automatically becomes an APL variable and can be manipulated by APL functions both in the body of a query and outside it, as in the following examples.

Q2. Sum the employees' salaries.

```
Q
TOTAL SALARY
```

SALARY belongs only to the relation EMP, so there is no trouble assigning this particular domain to its appropriate relation. The default options transform the query into:

```
Q
TOTAL (SALARY OF EMP)
WHEN
ALL EMP
```

This again produces a variable SALARY in the workspace, which could, for example, be an argument for the defined function AVERAGE.

```
Z ← AVERAGE SALARY
```

will give Z a value equivalent to the average salary of all the employees.

Queries may be nested one inside the other to any depth by using the function WITH to join together two or more values into a list of values in the condition list. (Actually any expression which evaluates to a two-dimensional matrix may be used, but a WITH expression is the easiest to grasp intuitively.) Thus the user may write queries like:

Q3. Names of employees working in departments located in New York or in Houston.

```
Q
(NAMES OF EMP)
WHEN
DEPT ISONEOF (DEPT NO OF DEPARTMENT)
              WHEN
              LOC ISONEOF NEWYORK WITH HOUSTON
```

ISONEOF is the function representing set inclusion. NAMES, incidentally, is being used as a synonym for NAME.

Comparison functions that perform scalar operations between corresponding elements in two ordered sets are also defined. These are useful when a "computed" domain has to be compared to an existing one, as in the following example.

Q4. Names of employees who earn more than their managers.

```
Q
(NAME OF EMP)
WHEN
SAL GT ((SAL OF EMP)
        WHEN
        CODE ISONEOF MANAGER) OWN MANAGER
```

Here the second (nested) query first finds the salaries of the managers. Then the function OWN builds a computed domain which has for every item in the domain MANAGER in EMP the corresponding salary. These values are then compared with the corresponding values of the domain SALARY in EMP to pick out the relevent names (if any).

Earlier we talked about constructing a path between relations. The necessity for this arises when the user wants attributes of one relation while imposing conditions on attributes of another. The burden of building such a path may be left to the system, as in the next example.

Q5. Names and commissions of employees in the department with the greatest sales.

```
Q
((NAMES WITH COMMISSION) OF EMP)
WHEN
(VOLUME OF SALES) EQ MAX VOLUME
```

The final VOLUME could be a domain either of SALES or of SUPPLY, so the user is asked to choose between the two. Once SALES has been specified, the system uses its inference algorithm to build a link between SALES and EMP. This is done by means of their common domain DEPT, and the query is restated as follows:

```
Q
((NAMES WITH COMMISSION) OF EMP)
WHEN
DEPT ISONEOF (DEPT OF SALES)
             WHEN
             (VOLUME OF SALES) EQ MAX (VOLUME OF SALES)
                                      WHEN
                                      ALL SALES
```

If the inference path should not be unique, the user is again offered a menu and asked to make a choice.

In this example there are two result variables, NAMES and COMMISSION.

AQL also provides facilities for "grouping", a way of representing many-to-one relationships.

Q6. Group the names of employees by their managers.

```
Q
(NAME OF EMP)
GROUPBY MANAGER
```

Quantifiers are also implemented as AQL functions. Their use is shown in the following example.

Q7. Departments which sell only items supplied by supplier 115.

```
Q
(DEPT OF SALES)
WHEN
DEPT HASONLY ITEM EQ (ITEM OF SUPPLY)
                     WHEN
                     SUPPLIER EQ 115.
```

The function EQ (and its companions GE, LT, etc.) performs the comparison between every ordered couple of elements of its arguments. HASONLY retrieves only those DEPTs all of whose occurrences in SALES are in correspondence only with some item supplied by 115.

When two domains of the same relation contain values extracted from the same set, they may be viewed as a set of ordered couples for which a certain predicate is true, i.e. as a binary relation in the algebraic sense (see, for example, [5]). Hence, operations like product, power, and transitive closure are applicable. Our final example illustrates this.

Q8. Code of the third level manager of employee number 117.

```
PWR 3
(MANAGER OF EMP)
WHEN
CODE EQ 117
```

The function PWR executes the query as many times as is specified in its right argument, substituting at each next step the constant with the result of the previous query. If the right argument of PWR is the empty vector, then transitive closure is executed. In our example this would mean that the code of the top level manager of employee 117 would have been obtained.

Sufficient has been said to give a general idea of what AQL looks like. More detailed discussion can be found in [6]. As can be seen, a reasonably experienced computer user would have little difficulty in formulating his questions using it. But it cannot be denied that a nonexperienced user would find it less easy. In the next section therefore we return to the problem of allowing natural language input.

5. NATURAL LANGUAGE TO FORMAL QUERY LANGUAGE TRANSLATION

Section 3 described the intermediate semantic representation established by the natural language analyzer and Section 4 the formal query language used to interrogate the data base. These form respectively the input and output structure for the translation phase described now.

Once again, description by means of an example offers the clearest mode of exposition. The example used is the sentence whose intermediate semantic representation was given in Section 3.

In order to establish the representation shown in Figure 1 the analysis routines used the semantic formulae given for each item in the vocabulary, and further semantic information mainly attached to the internal representation of prepositions and conjunctions. Now this representation has to be translated into the formal query language that interrogates the data base. The formal query language, as we have seen, consists of a set of predefined functions whose arguments are either the formal objects of the data base, i.e. the names of relations and domains, or embedded functions with the same type of argument. From this it is clear that the original lexical items of the question must be translated into the formal objects of the data base. In order to accomplish this we have added, for each word sense represented, a list of the domains and relations with which it may be associated. Thus, for example, the word "department", appearing in our example sentence, has related to it the list of possible data base associations shown in Table 4.

Table 4.

"department"....	(DEPT EMP
	DEPT SALES
	DEPT LOCATION
	---- DEPARTMENT)

This tells us that "department" may be associated with the domain DEPT of the relation EMP, the domain DEPT of the relation SALES, the domain DEPT of the relation LOCATION or may be directly connected with the relation DEPARTMENT. (It is not in the least necessary that all the domains with which a particular word is associated should have the same name: it simply helps the human

memory to call closely related domains by the same name.) We have called these lists "data base associations".

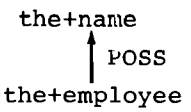
Thus, in essence, we have two levels of specification of the "meaning" of a word sense. The semantic formulae attempt to define its general meaning in the natural language used. Lists like that shown for "department" define its meaning within the restricted world of the data base.

Data base associations are heavily used by the translation algorithm. To see how, let us follow through the action of the algorithm on the example sentence given in Figure 1.

Consider first the root template, defined as that template which is not dependent through case ties on any other template --in our case, template 1 in Figure 2.

Figure 2.

Now we check the contents of the third position in the template. (We shall often call this the "object position" since templates are most easily conceptualized as actor-act-object-triples.) If the element in this position has a POSS link attached, as it does in the example, the whole structure is picked up and considered as a unit. Thus we have

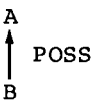


Immediately now the data base associations are used. First we look at those of the dependent. If there is any entry that refers directly to a relation (like the last entry in Table 4) it is assumed that this particular query refers to that relation. "Employee" has only one entry in its data base association, an association with the relation EMP, so this is taken as the request relation for this query. Moving now to the data base association for the element to which the POSS is attached, an association with this same relation EMP, via a domain NAME is found, so we can generate immediately

NAME OF EMP

as the first part of the formal query language representation.

This is the simplest case. The POSS link supplied the name of the relation, the element to which it was attached and the name of the domain in the same relation. Considerably more complicated situations are possible, which are worth discussing in some detail because they show that the addition of data base associations is an essential aid to translation. In order to facilitate discussion let us diagram the general structure of the object position with a POSS link as



Two basic situations are possible. The first of these is the situation we have just seen where B can refer directly to a relation. In that case, if A does not contain in its data base association an association with a domain of the same relation, there are a further two possibilities. Perhaps it has an empty data base association. In that case we take the master key of the relation specified by consideration of B. By master key is meant that domain or set of domains of the relation that uniquely identifies the elements of the relation, which is, in a sense, logically prior to the other domains. In the case of EMP, for example, it is CODE, the domain that contains the codes uniquely identifying each employee. So if the question had been "Give me a list of the employees..." where "list" has an empty data base association, the structure

CODE OF EMP

would have been generated.

Alternatively, A may have a nonempty data base association, but can contain an association with a domain in the relation specified by B. In that case we generate a structure with a domain specified by A, a relation specified by B, and leave it to the formal query language processing level to generate a logical path between the two. So, if the question was "Give me the location of the employee..." which, as far as the formal query language is concerned, is shorthand for "the location of the department of the employee..." we should simply generate

LOC OF EMP

which the formal query language processor would later transform into

```
(LOC OF DEPARTMENT)
WHEN
DEPTNO ISONEOF DEPT OF EMP
```

If B is not directly associated with a relation, it may be associated with what we have called a pseudo-relation. By this is meant a relation which can be defined in terms of "restriction", in Codd's sense [1], on another relation, and whose definition is permanently fixed. Thus, if the query had asked for "a list of managers", "managers" points us to a pseudo-relation defined as

```
(NAME OF EMP)
WHEN
CODE ISONEOF MANAGER
```

If A has an empty data base association, as would be the case with "a list of managers", this definition is simply lifted and inserted as the first part of the query. If A is associated with a domain, the structure generated is the domain specified by A of the relation specified in the pseudo-relation given by B. Once again the formal query language processor will supply a logical path from the first to the second.

The situation is considerably more complicated if B is associated only with a set of one or more domains. In this case, if A has an empty data base association, B is checked to see if it has already been used. If it has, there is no need to produce anything. This situation will only arise when there are multiple POSS links, as in "Give me a list of the names of the employees...". Since we always start with the lowest POSS, "names of the employees" will already have generated

```
NAME OF EMP
```

so "list of names" can be ignored.

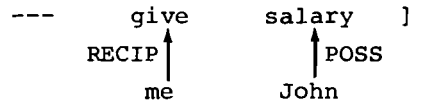
If B has not been used before, then its translation is the name of the domain with which it is associated, and the relation after the OF is found from the data base associations for B. So "Give me a list of the salaries..." will generate

```
SALARY OF EMP
```

If B is associated with a set of one or more domains and A also has a nonempty data base association, an attempt is made to find the strongest correlation between the two data base associations by forming the intersection of the relations in which the domains associated with A and B appear. If the intersection has only one member, that relation becomes the request relation for the query. This will arise, for example, with a query like "Give me the price of items..." where "price" is associated with the domain COST of SALES and "item" with (amongst other domains) the domain ITEM of the same relation SALES. From this is generated, after intersection has determined SALES as the appropriate relation,

```
(COST OF SALES)
WHEN
ITEM.......
```

By now the possibilities, as well as the reader, have been almost exhausted. If B is not a relation, not a pseudo-relation, and not a domain, we assume that it must be an item, i.e. a constant of the data base, something that can only be the value of a domain. If A is associated with a domain, we prefer it as the main part of the query, so that, for example, from "Give me John's salary" where the intermediate representation is



the query generated is

```
SALARY
WHEN
NAME EQ JOHN
```

where the information that "John" is the proper name of a person is picked up from a conventional semantic formula that is the same for all people's names.

If A is associated with a relation, the relation name is cross-checked with the possible relations attached to the formula of the item. If there is one relation that matches, this is preferred, and the master key of that relation is generated, followed by an automatic WHEN.

Twice now we have said "if the intersection has only one member" without specifying what happens if there is more than

one. In this case the query is genuinely ambiguous, and the user must be asked which relation he prefers.

So far generation from the root template has been considered, and, although the description is complex enough, not all possible cases have been covered. No mention has been made, for example, of what happens if the root position is empty, or if there is no POSS link, or if there are CONJ links to the root template--all not only possible but probable situations. But it is also clear that to continue description at this level of detail would be impossible. Fortunately the main point has been achieved and for the rest of the generation we can safely return to our example sentence and skate over other cases. The intention was to demonstrate that the data base associations were an indispensable aid to translation. It should be quite clear from the discussion so far that this is so.

To return to our example sentence. From template 1, the root template, we have

NAME OF EMP

("Give me", after a simple semantic check to make sure that it is an information seeking verb, is ignored.) To generate the rest of the query, the links to the root template are followed until a constant is found. In the example this means that we follow the APPOsitive link from template 1 to template 2, the SLOCAtive from template 2 to template 3 and the APPO from template 3 to template 4, where the first constant appears. Before generation starts from the template containing the constant we check whether other templates are linked to it by a CONJunction link. If such a linked template exists a tie is inserted from it to the root template if no link already exists. In our example template 5 is tied by a CONJ to template 4 but it is also already tied to template 1, so no extra link need be inserted.

Template 4 contains in its agent position an item that, via its data base associations, is linked to the domain ITEM of the relation SALES, and, furthermore, has "department" hanging from it by a POSS link. "Department" too is associated with the relation SALES, so that the first part of the conditional section of the request is determined. The verb position of template 4 is occupied by a comparison verb, the object position by the constant, so we can immediately generate

(ITEM OF SALES) EQ XYZ

Since we have moved away from the root template we automatically know that we are now generating the list of conditions part of

the query, so this new section can be joined to the section generated from the root template by a WHEN to give:

```
(NAME OF EMP)
WHEN
(ITEM OF SALES) EQ XYZ
```

As a simple security measure we now travel back up the links to the root template, checking each template that we pass to make sure that all the essential information for the formal query language representation has been dealt with. This primarily means checking the object position of each template for POSS links.

When the root template is reached, a link that has so far never been traversed, that from template 5, is found, This automatically means that the conditions so far generated should be enclosed in parentheses and a conjunction (AND, OR) generated. Then template 5 is dealt with, and by a process of reasoning similar to that used for template 4, the formal query language representation is completed, to give

```
(NAME OF EMP)
WHEN
((ITEM OF SALES) EQ XYZ
AND
(SALARY OF EMP) EQ 5000
```

6. CONCLUSION AND FUTURE PLANS

The system is, at present, unevenly developed. The description given here reveals that the part most in need of further work is the translation algorithm. There are some obvious gaps. Quantification, for example, is not yet dealt with at all. A new semantic primitive, QATTRIB, is planted in the intermediate semantic representation to mark where quantification occurs and will, eventually, trigger the application of a quantifier specialist during the translation phase. But the design and specification of this specialist has been left to one side for the moment on the grounds that it is a separate distinguishable problem that will not affect the overall design of the translation algorithm. A problem more intimately connected with overall design is that of sentences that contain no constant. Sometimes such a sentence presents no real problem. "List the names of all the employees", for example, consists only of a root template in the intermediate representation and the ordinary generation algorithm will produce a satisfactory formal query language representation. In other cases the problem is considerably more complex and requires much further work. However, even with the crude algorithm described in Section 5, the linguistic coverage obtained is quite wide

and is, we believe, at least equivalent to that of other existing natural language input data base systems.

The chief outstanding question with the natural language analyzer concerns its stability under a wide expansion of vocabulary. Wilks's orignial system had a larger vocabulary than any other primitives based system, but even so this only amounted to some 600+ items. We are now trying to establish a lexicon of 4000+ items. Preliminary results are encouraging.

The AQL interpreter and the RMS are already implemented. The next step here is to extend the RMS to include interlocks for multiuser shared access and update to the data base with authorization and recovery features. Checking for data consistency, especially during update operations, will also be provided. Data consistency will be defined by a set of integrity assertions about the domains of each relation in the data base.

Despite this long list of gaps in the system, the data base manipulation facilities already offered through the formal query language and the RMS are very powerful. The fact that any query that directly interrogates the data base creates an APL variable in the workspace allows the user to solve problems interactively with the system. The retrieval facilities are fast and efficient and can be used with a minimum of detailed specification. The user need have no fine knowledge of the structure of the data base and so need determine only what he wants in fairly general terms without having to give any instruction on how to get it.

Some real applications are running on the part of the system so far implemented. This should give some feedback which will allow an evaluation of the effectiveness of the system's facilities, especially as far as interactivity at the formal query language level is concerned.

REFERENCES

[1] Codd, E.F., A Relational Model of Data for Large Shared Data Banks, *Commun. ACM*, 13, 6 (1970), 377-387.

[2] Marcus, M., Diagnosis of a Notion of Grammar, in *Theoretical Issues in Natural Language Processing*, MIT, Cambridge, Massachusetts, 1975.

[3] Wilks, Y.A., An Intelligent Analyzer and Understander of English, *Commun. ACM*, 18, 5 (1975), 264-274.

[4] King, M., and Y.A. Wilks, *Semantics, Preference and Inference, A Full Description of a System and a Program*, ISSCO Report No. 18, Dec. 1976.

[5] Gries, D., *Compiler Construction for Digital Computers*, Wiley, New York, 1971.

[6] Antonacci, F., P. Dell'Orco, and V.N. Spadavecchia, AQL: An APL Based System for Accessing and Manipulating Data in a Relational Data Base System, in *Proceedings APL '76 Conference*, Association of Computing Machinery, New York, 1976.

The USL System for Data Analysis

H. Lehmann

INTRODUCTION

The use of natural language as a data manipulation language or, more generally, as a means of communication with the computer has been challenging many scholars. A number of experimental systems have been developed, and many different aspects of the problem have been addressed. Surveys of these systems can be found in [1,5,15,16,18]. When designing the USL system, the objectives were different in many respects from the experimental systems previously developed.

The USL system uses an independent data base management system (DBMS), and thus input sentences must be translated to the formal data manipulation language of the DBMS (a similar approach is also taken in the TORUS project [14]). Hence the main work to be done for the design and implementation of the present system was in writing a grammar for German that could be recognized by the parser (a modified form of Martin Kay's parser [4] also used in the REL system [17] and in the project at Karlsruhe [6]), and in developing suitable interpretation routines to perform the mapping from German to the data manipulation language.

The language of the USL system had to be defined in such a way that artificial restrictions in the use of the language could be avoided, because a language that looks natural in some respects, but behaves differently in others, will confuse the user and may be more difficult to learn than a formal language.

Although the most serious problems to be solved were problems of language, the main goal of the system is not the enhancement of the understanding of language but an attempt to find ways to bridge the gap between people and the computer, where by people we mean above all professionals whose interest is in their problems and not in the problems of electronic data processing. This main goal affects the manner in which language analysis can be done in the system, because it implies that linguistic information requested from the user, when he defines a new word for example, must be kept to a minimum. For the USL system this means that, for example, declension classes of nouns are not available, and it turned out that no serious problems arose from this restriction.

Concerning the semantics of natural language, there are several respects where the USL system uses a new or better

solution. The range of temporal expressions that can be interpreted is much wider than in the CHRONOS system [2] which was specifically designed for that purpose. Notoriously, the interpretation of quantifiers is a big problem in question answering systems, especially when more than one quantifier as well as negation are to be considered. For the purposes of the USL system, a thorough analysis of the scope of quantifiers in German sentences was done and an appropriate algorithm was implemented. The interpretation of coordinate noun phrases is an important problem, and it is also implemented in the system. Here, too, empirical investigations specific to German were necessary to be able to derive the required algorithm.

DESCRIPTION

System Overview

The USL system (see Figure 1) consists of:

- a dictionary,
- a set of syntax rules,
- a syntax rules compiler,
- a parser,
- an interpreter providing an interface to a DBMS, and
- processing routines for the manipulation of data.

The syntax of the language supplied to USL describes the conventions of a subset of German, the dictionary entries are function words like prepositions and conjunctions, and words whose meanings are constant over applications like names of months, and system commands. Names of concepts and the relationships between concepts differ from application to application and will be defined by the users according to the data and the intended use in the application. These identifier names can be taken from the set of German nouns, adjectives, and verbs (Maschine, männlich, wohnen, etc.). They can also be defined freely (e.g. X1, AVSAL).

The syntax is defined in modified Backus normal form (BNF). Each rule specifies a syntactic configuration to which the rule is applicable and specifies an output category that is to result after application of the rule. Associated with the rules are function calls representing the semantics of a given syntactic configuration with input elements as parameters. A set of rules in BNF--a grammar--is given as data to the syntax rules compiler, which converts the grammar into the format used by the parser and makes the language available for further use. This permits specification of user specialty languages and extensions to languages provided. For a detailed description of the grammar see [8,9,11,12].

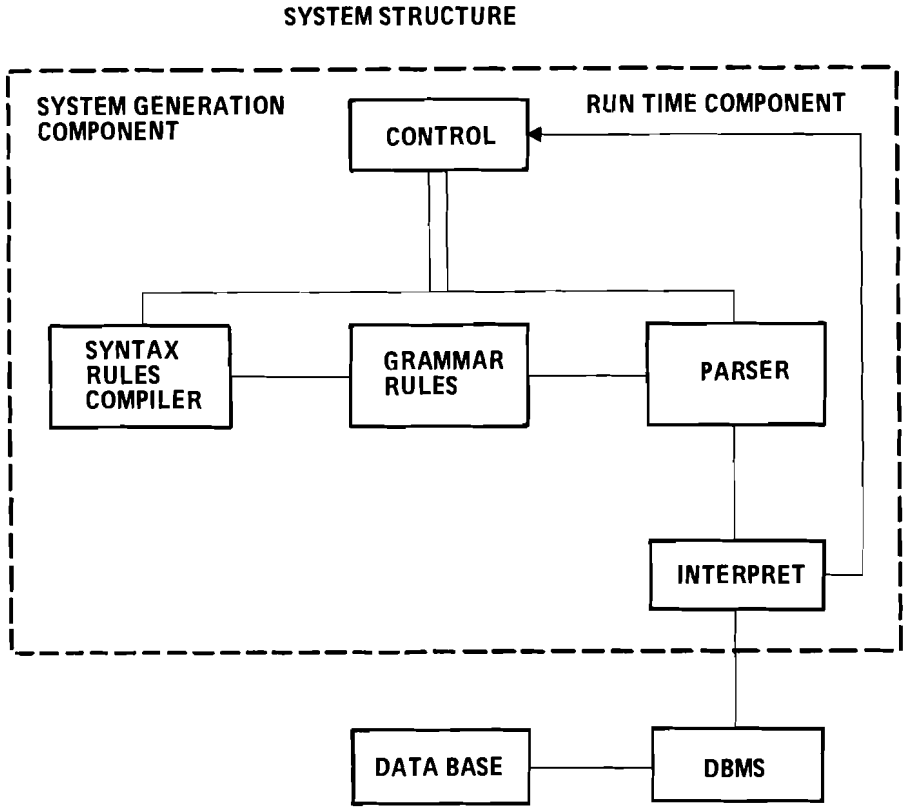


Figure 1. General system structure.

Input by a user is first analyzed by the parser, which builds up a tree structure that roughly represents the functional dependencies of the elements in the input string. The parser works bottom-up and from left to right. For ambiguous input all possible representations are built in parallel.

The tree structure built by the parser is passed to the interpreter which executes the function calls associated with each node of the tree. The functions successively build a normalized tree structure representing the dependency structure of the sentence except for the scope of quantifiers and pronominal references. This tree is then pruned to generate executable expressions in the formal data base (DB) language, which are then passed to the DBMS. Accordingly, the DBMS yields an answer to a query or performs an update function. A description of the semantic functions is given in [10].

A word not known to the language implemented is assumed to be a name. If the use of the word in the input string is

consistent with that assumption, the system will look for the required information and return an answer if the word is indeed a name and known in the DB, or it will signal that the information is not available. Otherwise, the system will signal that the input is not understood. The user may then use the definition facility to enter the word in the dictionary or repeat input if the word was misspelled. User errors are indicated by a set of diagnostic messages.

Semantic Model

We need a way to model the world in the environment of our system. Our model of the world is very simple, as it only consists of three kinds of entities, namely *objects*, *relations*, and *states*. Relations are sets of n-tuples of objects, where n is fixed for every relation. We introduce the notion of *semantic base* S as the pair ⟨U,R⟩, where U is a set of objects and R is a set of relations. We can now refine our model of the world by categorizing the objects of U, i.e. we introduce a set of *domains* D, where D is defined as a subset of the powerset of U. We can also refine the notion of relation by naming the places of each relation. These names we call *roles*. This gives us the possibility to classify the relations not only by degree but also by domains and roles. We can define a set of roles Ro from which every relation has to draw its roles.

The semantic model is conceived as a dynamic structure, i.e. we can imagine it to consist of states, where two states may differ with respect to U, R, D, and Ro. This concept of state is very similar to Carnap's state descriptions [3], or to the concept of "Zustand" in [7]. In order to be able to talk about the semantic model one obviously needs a language, which we shall call L. The language is also required to perform transitions from one state of the model to the other. A formal definition of the language L is given in [13].

Natural language reflects in its structure a common sense view of the world, here called the "linguistic world view". Here we have concepts like "thing", "property", "event", and many others. There are variations in what constitutes this world view from speaker to speaker and speech community to speech community. For a more detailed discussion of this situation see, for example, [7]. The USL world model like the linguistic world view is a model of the world, but it has a much simpler structure--as shown in Figure 2. The corresponding language L also is a very simple language--adequate to handle everything in the world model, but not more. In the USL system the simple model is used to interpret natural language, which of course implies that not everything that can be expressed in natural language can be interpreted by the system.

Theoretically, there are two ways to interpret natural language in a USL-like system (see Figure 2):

- Direct interpretation of the linguistic structure in terms of the model; and

- Translation of the linguistic structures into structures of a formal language, namely the formal manipulation language L, by means of a translation function t.

The second way was chosen in the USL system for several reasons, the most important of which was to be able to formulate a clear and clean interface to a data base system, i.e. a system that already performs the interpretation function i.

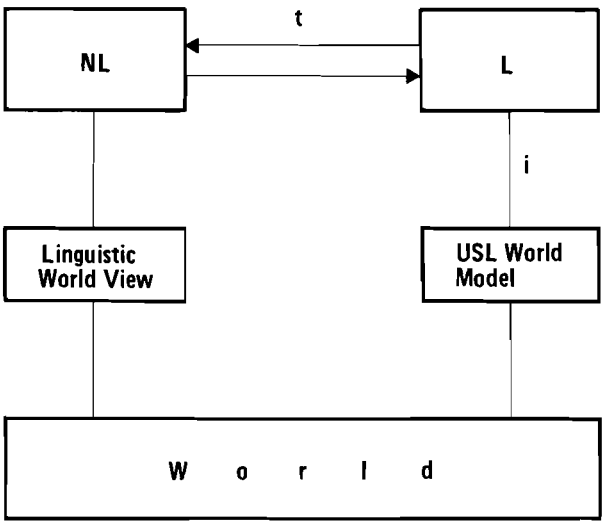


Figure 2. The USL interpretation of natural language (NL).

Words and Concepts

Concepts are expressed by words (and also by phrases) in natural language. The concepts can be represented in our model by relations, objects, or state transitions. The way in which a particular concept is represented depends on several things such as word class, utilization of the concept, taste, etc. When we talk about interpretation of words, we are dealing with three word classes: noun, adjective, and verb. Nouns can be subdivided into proper names and common nouns. Proper names will generally be represented as names of objects, and verbs as names of relations or state transitions (state transitions play only a minor role in our present context). Difficulties arise with common nouns and adjectives as they may sometimes be represented as objects and sometimes as names of relations. At least theoretically there is also the possibility to associate them with domain names or role names. For example:

SUPPLIER	
NAME	PRODUCT
JONES INC.	BOLTS

SUPPLIER, PRODUCT, and BOLTS are all common nouns. Note that in an organization as in the example, the question "Are bolts a product of Jones Inc.?" cannot be asked, unless either PRODUCT is also the name of a relation or a link exists somewhere from PRODUCT to SUPPLIER.

For a particular data base application, it will have to be decided whether each concept that occurs is to be treated as a name of an object or of a relation. In the USL system the choice was taken that words can only refer to either object names or relation names. As a consequence *standard role names* were introduced to refer to particular positions of a relation. These role names were defined with respect to cases governed by a verb and by prepositions governed by a verb or noun, and also by types of adverbials. The standard role names are:

NOM	nominative, first position of noun or adjective
ACC	accusative
DAT	dative
GEN	genitive
VON	genitive attribute
LA	place
LO	origin
LG	goal
LD	distance
LP	path
TA	point in time, date
TO	beginning
TG	end
TD	time interval
⟨preposition⟩	preposition governed by noun or verb .

The introduction of standard role names has the advantage that questions like

```
Who is a supplier of bolts?
?Sx [SUPPLIER(NOM=x ∧ VON='BOLTS')]
```

can be asked as well as

Who is a supplier of the product bolts?
?Sx[SUPPLIER(NOM=x∧VON='BOLTS')∧PRODUCT(NOM='BOLTS')]

In the USL system there is also a set of *standard domains* that is used. As a calculus of domains does not exist in our DBMS, only the most general domains are used. One place relations are used instead to classify the objects in the universe of discourse. The standard domains are:

ZAHL	number
WORT	word (character string)
DATUM	date, time of day
CODE	numeric code .

Sometimes a string of words (phrase) is taken to constitute a single concept, e.g. "data base administrator".

Syntactic Constructions and Their Interpretation

There are a set of constructions that are used in natural language queries and will have to be accounted for in any system that deals with natural language. They are more or less fundamental structures that exist in many languages:

⟨adjective⟩ ⟨noun⟩
⟨noun⟩ of ⟨noun phrase⟩
⟨noun⟩ ⟨noun⟩
⟨noun⟩ ⟨adverbial⟩
⟨noun⟩ greater than ⟨noun phrase⟩
⟨noun⟩ ⟨relative clause⟩
⟨noun⟩ and ⟨noun⟩
⟨quantifier⟩ ⟨noun⟩
⟨verb⟩ ⟨noun phrase⟩
⟨verb⟩ ⟨prepositional phrase⟩
⟨sentence⟩ ⟨adverbial⟩

Repeated occurrence, nesting, and overlap of these constructions leads to the phenomenon of structural ambiguity. Although structural ambiguity is handled by the parser of the USL system, it remains a problem in many cases, since parses that do not result in meaningful interpretations may be costly. Two cases of structural ambiguity occur quite often in German sentences:

- Nominative/accusative,
- Prepositional phrase or adverbial modifying either a noun or a verb.

For example:

```
Welches Alter hat Fritz?
Who left the room with the terminal?
```

Sentences like the above cannot be disambiguated syntactically, but only by the roles of the relations addressed. The treatment of most of the structures mentioned above is explained in the sequel; a fuller treatment can be found in [13].

⟨adjective⟩ ⟨noun⟩

These are constructions like "male employee". There are at least three ways to interpret this type of construction depending on the structures of the relations mapped to:

```
1.  CR EMPLOYEE(NAME,SEX)
    EMPLOYEE(NAME='BILL JONES' ∧ SEX='MALE')
2.  CR EMPLOYEE(NOM)
    CR MALE(NOM)
    CR FEMALE(NOM)
    EMPLOYEE(NOM='BILL JONES') ∧ MALE(NOM='BILL JONES')
3.  CR MALEEMPLOYEE(NOM)
    CR FEMALEEMPLOYEE(NOM)
    MALEEMPLOYEE(NOM='BILL JONES')
```

The representation that is chosen influences the types of questions that can be asked, and therefore an appropriate choice is important. The third representation may be desired for concepts like "relative humidity", "former customer", or "maximum capacity".

⟨noun⟩ of ⟨noun phrase⟩

This construction presupposes relations that have a VON-role, for example:

```
MANAGER(NOM,VON)
Who is the manager of Bill?
?Sx [MANAGER(NOM=x ∧ VON='BILL')]
```

There is an important relationship between genitive attributes--the kind of construction currently discussed--and the auxiliary verb "have". So one can also ask the above question in the form:

Which manager does Bill have?

or

Who does Bill have as a manager?

Therefore provision must be taken that questions of this form are interpreted like the ones having a genitive attribute in them.

A similar relationship exists with constructions containing the preposition "with":

Who are managers with more than 5 employees?
Which managers have more than 5 employees?
Who is a manager of more than 5 employees?
?Sx(>5)y[MANAGER(NOM=x∧VON=y)∧EMPLOYEE(NOM=y))]

⟨noun⟩ ⟨adverbial⟩

Adverbials comprise adverbials of time and place. When an adverbial modifies a noun, this presupposes that the relation corresponding to the noun has an appropriate role. Thus "income in 1975" is interpreted as

INCOME(NOM=x ∧ TA='1975')

"income from 1970 to 1975" corresponds to

(INCOME(NOM=x ∧ TA=y) ∧ y>='1970' ∧ y<='1975') .

⟨noun⟩ greater than ⟨noun phrase⟩

An example for this type of construction is

income greater than 5000
INCOME(NOM>5000 ∧ VON=x)

⟨noun⟩ ⟨relative clause⟩

Relative clauses are interpreted as restrictive relative clauses only, since it does not make much sense to add new information in a question. Relative clauses are interpreted

recursively, since they may be embedded. A simple example for this type of construction is

employee, who is experienced,
EMPLOYEE(NOM=x) ∧ EXPERIENCED(NOM=x)

which corresponds also to the interpretation of "experienced employee".

⟨noun⟩ and ⟨noun⟩

Constructions of this kind are interpreted in three different ways by the USL system:

- Conjunction,
- Sequence of displayed tables, and
- Combination of displayed tables.

Examples are:

Who is the manager of Jones and Smith?
Who are the managers and the employees?
What is the age and salary of employees of Jones?

The criteria for the determination of these cases are rather involved, and will not be presented in detail. It should be noted, however, that the interpretations are well defined in most cases, contrary to the widespread belief that people using "and" or "or" in natural language sometimes mean the one and sometimes the other, and that therefore natural language is not logical.

⟨quantifier⟩ ⟨noun⟩

Syntactically, a quantifier modifies a noun. However, the scope of the quantifier usually is a clause or sentence. Hence, quantifiers cannot be interpreted at the time they are recognized but only after the whole clause is processed. This is relatively simple as long as only one quantifier is present, but when there are more than one, scope ambiguities come into play. Scope ambiguities compete with preferred readings, which although not completely ruling out some readings definitely stress a single one. Preferred readings depend on word order (topicalization), subject versus nonsubject position of the quantified noun, and several other aspects.

The following quantifiers are treated by the present system:

jeder	every (each)
kein	no
alle	all
einige	some (any)
etliche	some
der	the
ein	a (one)

"Jeder" and "alle" are interpreted in the same way, although this is not always correct (compare "all men are equal" with "each man is equal"). In addition, numeric quantifiers are treated (e.g. "5 managers"). Since contextual (anaphoric) reference is not treated by the USL system, no distinction is made between definite and indefinite quantifiers.

⟨verb⟩ ⟨noun phrase⟩, ⟨verb⟩ ⟨prepositional phrase⟩

Verbs are interpreted as relations; in the simplest case, the noun phrase is a proper name, e.g.:

```
employ Bill Jones
EMPLOY(NOM=x ∧ ACC='BILL JONES')
```

When the noun phrase is to be interpreted as a relation also, like in

```
sell a computer
SELL(NOM=x ∧ ACC=y) ∧ COMPUTER (NOM=y)
```

a slightly more complex representation is required.

Noun phrases are conceived as arguments of the verb relation. At parsing time they are collected one by one, then they are translated recursively--noun phrases may have arguments of their own--into the corresponding formal expressions.

DATA ANALYSIS

Applications and Users Addressed

USL is designed as a problem solving system. In this context, problem solving means the creative process of performing nonroutine, nonrepetitive operations to solve a problem. The solution criteria of such problems are often incompletely defined, therefore several solution methods may be possible. Problem solvers are professionals in their field of specialty. They are professionals in administration and research who want to access data for decisionmaking easily and directly. They are used to explore different strategies and to work towards problem solution by successive approximation. USL is designed to help them to retrieve, update, and manipulate computer-stored data in their terminology, thus eliminating the need for formal data processing education. It can be easier for planners and decisionmakers to evaluate alternative solutions to a given problem and to understand consequences of decisions before they are implemented. Scientists can make use of USL to explore dependencies between data and to test hypotheses. This will not necessarily reduce the time needed for problem solution, but it can reduce the time spent adjusting the solution to facts that seemed unrelated at first.

Exploration and exploitation of natural resources is an application that uses geographical, geological, atmospheric, and agricultural data. The task includes examining locations of resources and determining exploitation profitability. The latter depends on many factors: availability of manpower, transportation cost, cost of refinement of raw materials, etc. USL contributes to solve these problems by providing the capabilities required to query data accumulated according to varying criteria and to evaluate different possible solutions.

Empirical research in industry or universities involves varying amounts of data depending on the project and the size of the group participating in the project. Aside from repetitive, routine operations, processing of these data is poorly supported. Studies in the humanities, for example in sociology and psychology, often require data collection from questionnaires. Many users would be served by small data bases, but they are not created today, because their size does not justify the cost of installation and maintenance by data processing personnel. Such DBs can be set up through USL and can improve the effectiveness of the people using them.

Data Base Design and Vocabulary Definition

Designing a DB means modeling a section of the world in terms of the data model of a given DB system. Data are collected in a variety of forms, such as tables, matrices, maps, networks, curves, lines, texts, or pictures. Not all of these forms are

equally well analyzed; nor are they equally well accepted by a DB system. Thus when an application is to be implemented, a first step is usually a conceptual analysis of the data available or required. Then a format has to be found for the data that can be understood by the system. After an appropriate format for an application has been developed, it can be used regardless of the availability of a computer or DB system. When an application is to be developed from scratch, an analysis of the structure of the data is required, i.e. the concepts that are to exist in the system have to be determined, and dependencies between the concepts have to be established also. It has further to be determined what kinds of results will be expected by the future users of the system. Part of the conceptual analysis consists in finding out how the future users are going to refer to the concepts to be implemented in their language, and the words have to be defined accordingly, and they have to be related to the relation names in the DB.

In the USL system a mechanism is provided to make vocabulary definition as easy as possible. When a new word is to be defined, it must be ensured that all columns of the respective relations can be reached by some formulation. Here a special problem occurs for USL and similar systems: the identification of concepts with relations with a number of roles and domains. This problem has been addressed at several places in this document (see the sections on words and concepts and on syntactic constructions in particular). Sometimes the same role name could be used for different domains, like in

```
manager of Bill Jones
manager of IBM  .
```

Although it might not cause problems even in this case, when the two domains in question were joined, one would still hesitate to do so in practice.

Use

Queries can be formulated as yes/no-questions, wh-questions, or commands using the verb "list". The answer to a wh-question or a command is a table that may be empty (if no object having the specified search criteria exists). Intermediary results can be stored in variables, e.g.

```
pm2 = the countries whose pmregion is 2  .
```

New data can be inserted into existing relations using declarative sentences, e.g.

```
John is the manager of Bill.
John is the manager of all married employees.
```

Deletions can only be performed by using the formal data manipulation language.

Basic arithmetic operations are available for scalars and row-wise application to relations that have a common "von"-column, e.g.

```
inc=salary+(salary/10)   .
```

In addition to these, column operations are available to compute sum, average, maximum, and minimum. There are also functions to convert dates and to compute time intervals. These basic operations can be used to define new functions, e.g.

```
square(x)='x**2'
list square(salary)   .
```

Functions that require specification of an algorithm can be added with the help of a programmer. No attempt is made to use natural language for automatic programming.

REFERENCES

[1] Batori, S., et al., *LIANA - Ein deutschsprachiges Frage-Antwort-System*, 1975.

[2] Bruce, B.C., A Model for Temporal References and Its Application in a Question Answering Program, *Artificial Intelligence*, 3, 1 (1972).

[3] Carnap, R., *Meaning and Necessity*, Chicago University Press, Chicago, 1947.

[4] Kay, M., *Experiments with a Powerful Parser*, presented at the Second International Conference on Computational Linguistics, Grenoble, August 1967.

[5] Kogon, R., et al., The User Specialty Languages System, in *Proceedings of the 6th Annual Meeting of the GI*, Berlin, 1976.

[6] Krägeloh, K.-D., and P.C. Lockemann, Hierarchies of Data Base Systems: An Example, *Information Systems*, 1 (1975).

[7] Lehmann, H., *Linguistische Modellbildung und Methodologie*, Max Niemeyer, Tübingen, 1973.

[8] Lehmann, H., and M. Zoeppritz, *Language Facilities of USL/German, Version II*, TN 75.01, Heidelberg Scientific Center, Heidelberg, 1975.

[9] Lehmann, H., and M. Zoeppritz, *Grammar Rules for German, Version II*, TN 75.02, Heidelberg Scientific Center, Heidelberg, 1975.

[10] Lehmann, H., and N. Ott, *Interpretation Routines for German Grammar Rules*, TN 75.03, Heidelberg Scientific Center, Heidelberg, 1975.

[11] Lehmann, H., and M. Zoeppritz, *Partition of German Grammar*, TN 75.05, Heidelberg Scientific Center, Heidelberg, 1975.

[12] Lehmann, H., and M. Zoeppritz, *Grammar Rules with Examples*, TN 75.06, Heidelberg Scientific Center, Heidelberg, 1975.

[13] Lehmann, H., The USL System - Its Objectives and Status, in *Proceedings of IBM Int. Technical Conf. on Relational Data Base Systems*, IBM, Bari, 1976.

[14] Mylopoulos, J., et al., TORUS - A Natural Language Understanding System for Data Management, presented at the 4th IJCAI, Tbilisi, August 1974.

[15] Simmons, R.F., Natural Language Question-Answering Systems, *Commun. ACM*, 13, 1 (1970).

[16] Sparck Jones, K., and M. Kay, *Linguistics and Information Science*, New York, 1973.

[17] Thompson, F.B., P.C. Lockemann, B.H. Dostert, and R.S. Deverill, REL: A Rapidly Extensible Language System, in *Proceedings of the 24th National ACM Conference*, Association of Computing Machinery, New York, New York, 1969.

[18] Walker, D.E., Automated Language Processing, in Carlos A. Cuadra, ed., *Annual Review of Information Science and Technology*, Vol. 8, AFIPS, Washington, D.C., 1973.

Appendix. Sample Session.

```
usl
USL --- VERSION: NOVEMBER 11, 1976
MELDUNGEN  - MESSAGES   - MELDINGEN     - MENSAJES
DEUTSCH(1) - ENGLISH(2) - NEDERLANDS(3) - ESPANOL(4)
META:
ENGLISH AVAILABLE FOR EDITING

:
?
start iiasa,demo
?
what is the base?

 BASE
 ------------------------------------------------------------
 (ROPERREQ+RCONREQ)
 RCOUNTRY
 RCONREQ
 RCLTIME
 RFAC
 ROPERREQ
 RPRODDEP
 RRESERVE
 RRESOURC
 RREFERENC*RPUBLICA;C
 RREF
 RWATERPC
?
whose base is 'rfac'?

 RELATION
 ------------------------------------------------------------
 ACT
 CAPACITY
 FACILITY
 FACNAME
 FACCLASS
 INRESQT
 NEED
 OUTRESNO
 OUTRESQT
?
what are terms for 'inresqt'?
```

```
 TERM
 ------------------------------------------------------------
 INRESQT
?
list country 70

 COUNTRY                                   NAME
 ------------------------------------------------------------
         70                                FEDERAL REPUBLIC OF GERM
?
what is the pmregion of 70?

 PMREGION
 ------------------------------------------------------------
          2
?
list the countries whose pmregion is 2

 COUNTRY                                   NAME
 ------------------------------------------------------------
         58                                TURKEY
         60                                NETHERLANDS
         61                                YUGOSLAVIA
         63                                AUSTRIA
         64                                BELGIUM
         67                                DENMARK
         68                                FINLAND
         69                                FRANCE
         70                                FEDERAL REPUBLIC OF GERM
         72                                GREECE
         74                                ITALY
         75                                NORWAY
         77                                PORTUGAL
         79                                SPAIN
         80                                SWEDEN
         81                                SWITZERLAND
         82                                UNITED KINGDOM
        102                                ANDORA
        138                                ICELAND
        139                                IRELAND
        151                                LIECHTENSTEIN
        152                                LUXEMBOURG
        154                                MALTA
        157                                MONACO
        174                                SAN MARINO
        191                                VATICAN CITY STATE

?
what are comments on reference 1?

 COMMENT
 ------------------------------------------------------------
 MAJOR OIL FIELDS AROUND THE WORLD
```

```
?
what is the reqired quantity for the construction of 101?
NOT UNDERSTOOD: THE FOLLOWING WORDS WERE ASSUMED TO BE PROPER NAMES:
 REQIRED
?
what is the required quantity for the construction of 101?

   INTERPRETATION        1
 QTY
 ------------------------------------------------------------
          0
          1,080000
        500
       2600
       4400
       4600
      27800
      80500

   INTERPRETATION        2
 OBJECT REQUESTED DOES NOT EXIST
?
list the required quantities for the construction of 101

   INTERPRETATION        1
 OBJECT REQUESTED DOES NOT EXIST

   INTERPRETATION        2
 RESOURCE          QTY                 REF                 PURPOSE
 ------------------------------------------------------------------
         52             4400                  10                1101
         70                1,080000           10                1101
         71                0                  10                1101
         81            27800                  10                1101
         82            80500                  10                1101
         83             2600                  10                1101
         84             4600                  10                1101
         85              500                  10                1101
?
end
```

A Natural Language Interface Facility and Its Application to a IIASA Data Base*

G.G. Hendrix

INTRODUCTION

This note describes LIFER, a practical facility for creating natural language interfaces to other computer software. Emphasizing human engineering, LIFER has bundled natural language specification and parsing technology into one convenient package. This package includes an automatic facility for handling elliptical (i.e. incomplete) inputs, a spelling corrector, a grammar editor, and a mechanism that allows even novices to extend the language recognized by the system through the use of paraphrase. Offering a range of capabilities that supports both simple and complex interfaces, LIFER allows casual users to rapidly create workable systems while giving ambitious users the tools needed to produce powerful and more efficient language definitions. Experience with the system has shown that for some applications, very practicable interfaces may be created in a few days. In particular, an interface to an example set of IIASA data was created in only two days. The resulting systems are directly usable by business executives, researchers, and office workers whose areas of expertise lie outside the field of computer science.

THE LIFER SYSTEM

LIFER is composed of two basic parts: a set of interactive language specification functions and a parser. In standard practice, a system builder uses the language specification functions to define an application language. This application language is a subset of a natural language (e.g. English) that is appropriate for interacting with an existing software product. Using this language specification, the LIFER parser will then interpret natural language inputs, translating them into appropriate interactions with the application software.

Example interactions with a LIFER application language for a data base access system are presented in Table 1. This particular language definition was developed for a collection of IIASA

---

*The work reported herein was conducted under Stanford Research Institute's Proprietary, Internal Research and Development Program.

Table 1. Example interactions with LIFER.

```
-What is the depth of the golden spike deposit?
 PARSED!
 (DEPO.NO 29 DEPTH 1786.4)

-number of wells and API?
 Trying Ellipsis:  WHAT IS THE NUMBER OF WELLS AND API OF THE GOLDEN
 SPIKE DEPOSIT
 (DEPO.NO 29 WELLS 21 API 36.0)

-of bell creek
 Trying Ellipsis:  WHAT IS THE NUMBER OF WELLS AND API OF BELL CREEK
 (DEPO.NO 308 WELL 250 API 32.0)

-deposit 1
 Trying Ellipsis: WHAT IS THE NUMBER OF WELLS AND API OF DEPOSIT 1
 (DEPO.NO 1  WELLS 63 API 38.0)

-deposits in the Union of Soviet Socialist Republics
 Trying Ellipsis:  WHAT IS THE NUMBER OF WELLS AND API OF DEPOSITS
 IN THE UNION OF SOVIET SOCIALIST REPUBLICS
 (DEPO.NO  474 WELLS    2000 API 27.2)
 (DEPO.NO  475 WELLS      10 API UNKNOWN)
 .
 .
 .
 (DEPO.NO 1128 WELLS UNKNOWN API UNKNOWN)
-What is the average API gravtiy and depth for American oil deposits
           spelling -> GRAVITY
 PARSED!

 API
 233 items accepted
 Average is 29.63305

 DEPTH
 236 items accepted
 Average is 1265.308

-maximum
 Trying Ellipsis:  WHAT IS THE MAXIMUM API GRAVITY AND DEPTH FOR
 AMERICAN OIL DEPOSITS

 API 57.0
 DEPTH 5486.4

-what oil deposits in the U.S.A. were discovered in 1969
 Trying Ellipsis:  ELLIPSIS HAS FAILED
 THE PARSER DOES NOT EXPECT THE WORD
 "U.S.A." TO FOLLOW
 "WHAT OIL DEPOSITS IN THE".
  OPTIONS FOR NEXT WORD OR META SYMBOL ARE: <COUNTRY>
```

Table 1. (cont'd)

```
-How is the symbol <COUNTRY> used
 PARSED!
 <COUNTRY> may be any member of the set {AFGANISTAN ALBANIA ...
 ... USA ... ZAMBIA}
 <COUNTRY> may be any sequence of words following one of the patterns:
 <COUNTRY> => CAPE VERDE ISLANDS
           => CENTRAL AFRICAN REPUBLIC
           .
           .
           .
           => UNITED STATES OF AMERICA

-use USA for U.S.A. in -2
 PARSED!
 (DEPO.NO 212)
 (DEPO.NO 342)
 (DEPO.NO 436)

-What is the resource of the OKC deposit
 Trying Ellipsis:  ELLIPSIS HAS FAILED
 THE PARSER DOES NOT EXPECT THE WORD
 "OKC" TO FOLLOW
 "WHAT IS THE RESOURCE OF THE".
  OPTIONS FOR NEXT WORD OR META SYMBOL ARE:  <DEPOSIT>

-let (What is the resource of the OKC deposit) be a paraphrase of
 (What is the resource of the Oklahoma City deposit)
 PARSED!
 MAY LIFER ASSUME THAT "OKC" MAY
 ALWAYS BE USED FOR "OKLAHOMA CITY"?
 (TYPE YES OR NO)
 yes
 <D-NAME> => OKC

-what is the resource of the OKC deposit?
 PARSED!
 (DEPO.NO 336 RESOURCE OIL)

-Give the discovery date for OKC
 PARSED!
 (DEPO.NO 336 DISC.DATE 1928)
-Show geological type Salem
 Trying Ellipsis:  ELLIPSIS HAS FAILED
   [error message is printed.]

-Let (show Geological type Salem) be a paraphrase of (What is the
 geological type of the Salem deposit)
 PARSED!
 LIFER.TOP.GRAMMAR  =>  <WH/LIST><ATTRIBUTES><DEPOSIT>

-Show geological type Salem
 PARSED!
 (DEPO.NO 242 PAY MISS)
```

Table 1. (cont'd)

```
-Country country number and production
 Trying Ellipsis:  SHOW COUNTRY COUNTRY NUMBER AND PRODUCTION SALEM
 (DEPO.NO 242 COUNTRY (UNITED STATES OF AMERICA) COUNTRY.NO 31
 QUANTITY 3.0)

 [now try another compact input similar to SHOW GEOLOGICAL TYPE
 SALEM]
-Print number of wells Soviet Oil deposits with API over 37.5
 PARSED!
 (DEPO.NO 476 WELLS UNKNOWN API 41.0)
 (DEPO.NO 481 WELLS     500 API 38.0)
 .
 .
 .
```

energy data with only two day's work. The system user types in a query or command in ordinary English, followed by a carriage return. The LIFER parser then begins processing the input. When analysis is complete, LIFER types "PARSED!" and invokes application software (here, a data management system) to respond.

An important feature of the LIFER parser is an ability to process elliptical (incomplete) inputs. Thus, if the system is asked

WHAT IS THE DEPTH OF THE GOLDEN SPIKE DEPOSIT?

then the input

OF BELL CREEK

will be interpreted as WHAT IS THE DEPTH OF BELL CREEK. Analysis of incomplete inputs is performed automatically by LIFER, making it unnecessary for the system builder to explicitly define elliptical constructions in the application language.

If a user misspells a word, LIFER attempts to correct the error using the INTERLISP spelling corrector [3]. If the parser cannot account for an input in terms of the application language definition, user-oriented error messages are printed that indicate what LIFER was able to understand and that suggest means of correcting the error.

The definer of the language interface can intermix calls to LIFER that extend or modify the language definition with calls to the parser that utilize the developing language system. This aids system builders in the task of defining the application

language, allowing them to operate in a rapid extend and test mode. Perhaps more importantly, it provides the basis for a mechanism through which naive users may extend their language by employing easy to understand notions such as synonyms and paraphrases. Provisions may be included in the application language for interfacing with LIFER's own language specification functions, allowing users to give natural language commands for extending the language itself. This is illustrated by the paraphrase examples of Table 1.

The LIFER parser uses an augmented, finite state transition network [4]. The LIFER language specification functions construct these underlying transition networks automatically from language production rules to the type commonly used by both linguists and compiler builders. The production rules may be easily modified and tested interactively, allowing sophisticated language definitions to be produced within a short period of time.

In using LIFER, interface builders typically embed considerable semantic information in the syntax of the application language. For example, words like JOHN and AGE would not be grouped together into a single ⟨NOUN⟩ category. Rather, JOHN would be treated as a ⟨PERSON⟩, and AGE as an ⟨ATTRIBUTE⟩. Similarly, very specific sentence patterns such as

WHAT IS THE ⟨ATTRIBUTE⟩ OF ⟨PERSON⟩

are typically used in LIFER instead of more general patterns such as

⟨NOUN-PHRASE⟩ ⟨VERB-PHRASE⟩ .

For each syntactic pattern, the interface builder supplies an expression for computing the interpretation of instances of the pattern. Expressions for sentence-level patterns usually invoke application software to answer questions or carry out commands.

Example interactions defining a LIFER application language are shown in Table 2. First, application information concerning biographic data is stored on property lists for later querying. Then the function MAKE.SET is called to define some word/phrase categories. The category ⟨ATTRIBUTE⟩, for example, is defined to include such words as AGE and OCCUPATION. Next, function PATTERN.DEFINE is used to add the productions

⟨ATTR-SET⟩ => ⟨ATTRIBUTE⟩

and

⟨ATTR-SET⟩ => ⟨ATTRIBUTE⟩ AND ⟨ATTR-SET⟩

Table 2. Defining an application language.

```
   {set up data to be queried}
-SETPROPLIST(JEWELL.FLEMING (AGE 35 OCCUPATION TEACHER HEIGHT 5.5
                             WEIGHT 105))
-SETPROPLIST(IVAN.FRYMIRE   (AGE 40 OCCUPATION FARMER HEIGHT 6.2
                             WEIGHT 225))

   {MAKE.SET and PATTERN.DEFINE extend the language definition}
-MAKE.SET(<PERSON> (JEWELL.FLEMING IVAN.FRYMIRE ...))
-MAKE.SET(<ATTRIBUTE> (AGE OCCUPATION HEIGHT WEIGHT))
-MAKE.SET(<IS/ARE> (IS ARE))
-PATTERN.DEFINE(<ATTR-SET> (<ATTRIBUTE>)
                (LIST <ATTRIBUTE>))
-PATTERN.DEFINE(<ATTR-SET> (<ATTRIBUTE> AND <ATTR-SET>)
                (CONS <ATTRIBUTE> <ATTR-SET>))
-PATTERN.DEFINE((WHAT <IS/ARE> THE <ATTR-SET> OF <PERSON>)
                (MAPCONC <ATTR-SET> (FUNCTION (LAMBDA (A)
                    (LIST A (GETPROP <PERSON> A))))))

   {a call to LIFER.INPUT sends subsequent inputs to the parser}
-(LIFER.INPUT)

   {start NL interactions using grammar defined above}
-what is the occupation of jewell.fleming
PARSED!
(OCCUPATION TEACHER)
-age and weight
TRYING ELLIPSIS:  WHAT IS THE AGE AND WEIGHT OF JEWELL.FLEMING
(AGE 35 WEIGHT 105)

   {MAKE.SET is called to add variety to persons' names}
   {leading ! sends line to LISP'S EVAL, instead of to parser}
-!MAKE.SET(<PERSON> ((JEWELL . JEWELL.FLEMING)
                     (IVAN . IVAN.FRYMIRE)
                     ((JEWELL FLEMING) . JEWELL.FLEMING)
                     ((IVAN FRYMIRE) . IVAN.FRYMIRE))

   {now more English input}
-what is the height of ivan frymier
 (assumed spelling error)==>FRYMIRE
PARSED!
(HEIGHT 6.2)
-of jewell
TRYING ELLIPSIS: WHAT IS THE HEIGHT OF JEWELL
(HEIGHT 5.5)
   {define a paraphrase in English}
-define "give the height of ivan" like "what is the height of ivan"
PARSED!
LIFER.TOP.GRAM => GIVE THE <ATTR-SET> OF <PERSON>
   {output above shows LIFER's generalization of the paraphrase}
   {now try an input based on the paraphrase above}
-give the age and occupation of jewell fleming
PARSED!
(AGE 35 OCCUPATION TEACHER)
```

to the language definition, establishing an ⟨ATTR-SET⟩ as a list of one or more attributes separated by ANDs. The third call to PATTERN.DEFINE sets up a top-level sentence pattern of the form

WHAT ⟨IS/ARE⟩ THE ⟨ATTR-SET⟩ OF ⟨PERSON⟩

which can match such queries as

WHAT IS THE AGE AND OCCUPATION OF JEWELL.FLEMING .

The expression for computing the value of this query maps down the list of attributes that are sought and extracts their values from the property list of the ⟨PERSON⟩.

After the function LIFER.INPUT is called, all lines of input are sent to the LIFER parser for processing. The first query of the example is a complete sentence, but the second is elliptical. No special patterns are needed to deal with this elliptic query. A more complex use of MAKE.SET and examples of the spelling corrector are shown in later interactions in Table 2. Many other features are available, including a grammar editor, aids for processing anaphora, and a mechanism for using LISP predicates to define syntactic categories.

LIFER is implemented in PDP-10 INTERLISP, with the basic system requiring an additional 14K words above the 150K used by INTERLISP. An extensive language definition for communicating with a large data base (70 fields on 14 files with hundreds of records) requires an additional 30K, including some data base access routines. Such sentences as

WHAT IS THE DEPTH OF THE GOLDEN SPIKE DEPOSIT

parse in less than 0.2 s of CPU time on the DEC KL-10, faster than the sentences are usually spoken or typed.

For more information about how LIFER works and about how application languages may be defined, see [1] and [2].

REFERENCES

[1] Hendrix, G.G., *The LIFER Manual: A Guide to Building Practical Natural Language Interfaces*, Tech Note 138, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, 1977.

[2] Hendrix, G.G., *Human Engineering for Applied Natural Language Processing*, Tech Note 139, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, 1977.

[3] Teitelman, W., *INTERLISP Reference Manual*, XEROX Palo Alto Research Center, Palo Alto, California, 1975.

[4] Woods, W.A., Transition Network Grammars for Natural Language Analysis, *Commun. ACM*, 13, 10 (1970), 591-606.

Natural Language Processing Within A Restricted Context

V. Briabrin and G. Senin

INTRODUCTION

The Dialogue Information Logical System (DILOS) has been developed for serving as an "intelligent" mediator between the user and a set of applied programs and data modules [1]. The system is written in LISP and can be divided functionally into several parts called "processors". Three executive processors are activated by the "formal interface expressions" (ϕ expressions) and they perform logical inference, information retrieval, calculation planning, and control. Thus ϕ language constitutes one possible medium for user communication with the system.

The special Linguistic Processor (LINGP) is a front-end part of the system intended for the transformation of the input natural language phrases (NL phrases) into the corresponding ϕ expressions. We present here a short description of LINGP operation and its underlying principles. The system has been developed in Moscow for a BESM-6 computer and transferred to the PDP-11/45 at IIASA [2].

THE GENERAL PRINCIPLES OF NL → ϕ TRANSFORMATION

Let us consider LINGP interaction with the information retrieval processor (IRP). The general syntax of ϕ expression directed to IRP is as follows [2]:

$$\left(\langle\text{func-name}\rangle\ \langle\text{div-name}\rangle\ \left\{\begin{matrix}\langle\text{obj-names}\rangle\\ \langle\text{var}\rangle\end{matrix}\right\} : \left\{\begin{matrix}\langle\text{restriction}\rangle\\ \langle\text{prescription}\rangle\end{matrix}\right\} ;\cdots\right) . \tag{1}$$

⟨func-name⟩ defines the type of operation (FIND, ADD, DEL, ···). ⟨div-name⟩ sets up the scope of operation to be performed, i.e. establishes a current data base division name. ⟨obj-names⟩ put further restrictions on the scope of operation by requiring that it be applied only to the objects with the given names. If ⟨var⟩ is used instead of the ⟨obj-names⟩ then all objects of the current division participate in the operation. ⟨restriction⟩ is represented usually by the pair {⟨ind⟩ ⟨val⟩} implying that a required object should possess the given value ⟨val⟩ under the given indicator ⟨ind⟩. ⟨prescription⟩ is represented by the pair

{⟨ind⟩ ⟨var⟩} requiring the system to extract the value of ⟨ind⟩ property from a given object and assign it to the variable ⟨var⟩. ⟨ind⟩ could be represented by an atom or a list; ⟨val⟩ by an atom, a list, a number, an interval, or a set of values. ⟨var⟩ is a pattern variable designated by =⟨identifier⟩.

Example:

(FIND CITIES =X : LOC USSR ; POPUL 1.0 ; INDUST =Y)
div-name var restr-1 restr-2 prescr

In the process of NL → ϕ transformation an attempt is made to tackle each word from the input string in such a way that it would help to fill an appropriate position in the ϕ expression that becomes an output of LINGP. Thus a phrase "What industries are in the USSR cities with population 1.0?" is transformed into a ϕ expression as in the above example. For this purpose a problem-oriented vocabulary (part of the data base) should contain appropriate entries for the words form the input phrase. Each word has, among other properties, two that are the most essential:

- an internal code, substituting the given word in the constructed ϕ expression;
- a semantic type (S-type), designating the most likely role played by the given word in the process of NL → ϕ transformation.

The analysis is directed by the augmented transition network (ATN) where each node contains *preconditions* allowing transitions from one state to another and *predictions* about the likely S-types of the words that can occur in the current state. Preconditions could be connected with:

- features (properties) of the current input symbol, particularly its S-type;
- contents of the "registers" (variables) reflecting the history of input phrase processing.

Of course, an input phrase could contain "unknown" words not found in the vocabulary. A special arrangement is made for dealing with such words and we shall discuss it later.

RESTRICTED CONTEXT

The problem of natural language processing has always been considered difficult because it was believed that a processing system should operate successfully in a practically infinite context and handle a gigantic variety of individual lexicons.

However we accept the hypothesis that natural language communication with the computing system, intended for specific problem-solving, involves a rather restricted lexicon and context.

Two kinds of contexts are distinguished:

- The general context is defined by the pragmatics of the data base management, i.e. each input phrase tends to be converted into a meaningful ϕ expression leading to some operation on the data base contents;

- The local context is specified by the particular knowledge of the given problem area, and is represented by the set of class concepts, individual objects, relations facts, and rules of inference that reside in the current data base division (file).

Thus LINGP operates in the restricted environment defined by the general and specific local contexts. Moreover a local context is bound to and often created by an individual or a group of individuals who rarely change their style of conversation with the system after it has been established during the first 5-10 terminal sessions. Hence the system interpretation of the same words (reflected in their S-types) could be different when switching from one group of users to another but this fact does not create additional problems; on the contrary it alleviates LINGP operation by applying the same ATN mechanism to different problem-oriented knowledge.

ATOMIC S-TYPES AND THEIR COMBINATIONS

Each word possesses one of the following elementary (or atomic) S-types:

i - plays the role of ⟨ind⟩ in the ϕ expression;

v - plays the role of ⟨val⟩;

q - designates a query word;

p - a punctuation mark: such a word or character usually serves for transition to another state (changing expectations);

or - separator of alternatives;

leq, greq - substitutes for the words, designating "≤" and "≥" relations;

fn - plays the role of ⟨func-name⟩;

fl - plays the role of ⟨div-name⟩;

n - plays the role of ⟨obj-name⟩;

c - designates a "superconcept" of an object;

aa - designates an auxiliary action (i.e. calculation of minimum, maximum, average, etc.);

last - marks the end of the text.

Generally speaking, the S-type does not depend directly on syntactic properties of a word or its "ordinary" semantics, but it is entirely defined by the above mentioned contexts. For example, if we consider a local context describing the employees of an institution, then LINGP possibly has to deal with the following words and their S-types:

```
S-type ("get") = S-type ("salary") = i
S-type ("sit") = S-type ("room") = i
S-type ("earlier") = S-type ("before") = leq
```

Different types of correspondence exist between words and their senses (reflected in S-types). This could be a one-to-one correspondence, i.e. "1 word → 1 sense", but three other cases could also emerge.

Auxiliaries, i.e. words with "less than atomic" sense, are processed during the preediting stage (see below).

Composites, i.e. words with "more than atomic" sense, are assigned sequences of atomic S-types (in the form of LISP-lists). When encountering such a word the input string processing is suspended until all atomic senses constituting the composite sense are tackled in a proper way. Afterwards the input string processing is resumed. For example:

```
oldest = most + age → aa + i
who = what + person → q + c
woman = person + sex + female → c + i + v
```

Homonyms, i.e. words with multiple (alternative) senses s1,s2,••• , are represented by lists with the form: (, s1 s2 •••). We shall call it a "list of alternative senses" (LAS). Each homonym creates a branch point (BP) in the input string processing. All the necessary information is stored at the BP; then the first (the next) element from LAS is extracted and treated as a possible S-type of the current word. If the following processing becomes upset for some reason, then the analysis backtracks to the BP, restores all the necessary information, and tries to handle the next alternative from LAS.

Besides these three cases some words may be declared as "unimportant"; they are assigned "null" S-type and LINGP ignores them in the process of input string analysis (e.g. articles, some prepositions, etc.).

PROCESSING OF UNKNOWN WORDS

The system can deal with the unknown words in two modes.

In a "careful" mode LINGP asks the user about each unknown word, stores the information received in the temporary vocabulary (if the user encourages the system to do so), and proceeds with normal analysis.

In a "careless" mode all the unknown words are assigned "null" S-type requiring the system to ignore them at the first scan. The analysis proceeds from this moment exactly as in the case of homonyms, except that the LAS contents emerges not from the vocabulary but from the ATN state in which the corresponding unknown word was encountered. As has been mentioned each ATN state contains predictions about S-types that are "acceptable" in this state and these predictions become a source of LAS contents. Thus in a careless mode each unknown word creates a new BP leading the system to backtrack to this point if the analysis fails in the future.

When all the alternatives from LAS are exhausted and there is still no success (LINGP or the user is unhappy with the constructed ϕ expression) then we have two possibilities:

- to cease processing at the current BP (i.e. backtracking only one level, not more);
- to backtrack along the entire tree of alternatives with an attempt to consider all the possibilities created by homonyms and unknown words.

The user can help the system in the "careless" dealing with the unknown words. Instead of simply refusing to accept the system's ϕ interpretation of the original phrase, the user can make changes in the "working" vocabulary so that it contains only the words from the input phrase. In this case some BPs become excluded from the tree of alternatives thus speeding up the analysis.

NL PHRASE → ϕ EXPRESSION CORRESPONDENCE

Each NL phrase is mapped into exactly one output ϕ expression except for the following cases:

Preliminaries: execution of the "basic" expression is possible only after some preliminary actions.

Example: Who gets more than Brown?
Preliminary: How much does Brown get?
Basic: Whose salary is more than the one just found?

Additions: after execution of the basic expression some additional actions are necessary.

Example: What is the average salary of RDD laboratory employees?
Basic: What are the salaries of RDD laboratory employees?
Addition: Compute the average of the found numbers.

When these situations are recognized the corresponding ϕ expressions are generated and pushed into the special stores ("preliminary" and "addition"). After terminating the analysis, all preliminaries are executed first, then the basic expression is processed and after that all additions are executed.

PREEDITING STAGE

Words marked in the vocabulary as auxiliaries are processed before the principal block of translator starts operation. In the vocabulary the following information is connected with such words:

- What is the "master" of the given word to which it "adds" its sense;
- What is the "summary" sense of the two words.

As a rule, auxiliaries are predicates and syntactically govern their master. The master is usually represented at the entry by means of its syntactic and possibly semantic properties. If the master is found in the sentence it acquires the "total" sense. The auxiliary word in this case is excluded from the input string and does not participate in further considerations. The following NL word classes can be considered as auxiliaries:

- prepositions and postpositions;
- articles;
- elements used within analytical forms of verbs, adverbs, and adjectives;
- possibly, some adverbs, adjectives, etc.

Marking a word as an auxiliary is guided by consideration of convenience and uniformity of further analysis rather than for linguistic reasons and upon the whole is determined by the pragmatic purposes of the system.

IMPLEMENTATION

All programs are written in LISP*. The general and problem-oriented vocabularies as well as the ATN are stored in the external memory as special divisions of the "model data base". Thus the system in fact is not dependent on any specific NL and is easily convertible from one language to another.

The system is rather flexible because it can be adapted for special purposes by slight ATN amendment, and not by changing the basic programs. Particular vocabulary serves as a physical embodiment of some local context and the ATN of the general one.

The system transfer from the BESM-6 to the PDP-11/45 has required some notational amendments and modifications of input/output procedures. It has been tested experimentally and the performance encouraged us to further development and promotion. Examples of a user natural language communication with an experimental data base are given in the Appendixes.

REFERENCES

[1] Briabrin, V.M., and D.A. Pospelov, DILOS - Dialog System for Information Retrieval, Computation and Logical Inference, in *Conference on Artificial Intelligence: Question-Answering Systems*, CP-76-6, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

[2] Briabrin, V.M., *DILOS Reference Manual: Part 1*, RM-76-52, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

*The version of LISP translator for the BESM-6 computer was developed by S.S. Lavrov, G.S. Sylagadze, and V.M. Yufa.

Appendix 1.

```
"READY TO SPEAK"

=>what are the IIASA areas and programs , their leaders and budgets ?

"DOES THE FOLLOWING CORRESPOND TO THE INPUT PHRASE ?"

(find *IIASA =$nn : * resarea ; leader =$leader ; budg =$budg)

=>yes

EXECUTE?

=>yes

"START EXECUTION.  BE PATIENT"
```

FA	rabar	(8000 0)
EN	haefele	(10400 11210)
SDS	balinski	(12200 0)
MT	straszak	(5000 165)
HSS	hansen	(7000 1342)
RE	vasiliev	(7200 4182)

```
"CONTINUE SPEAKING ?"

=>yes

"READY TO SPEAK"

=>what areas have internal budgets between 7000 and 12000 ?

"DOES THE FOLLOWING CORRESPOND TO THE INPUT PHRASE ?"

(find *IIASA =$nn : * resarea ; (budg int) (: 7000 12000) =$int)

=>yes

EXECUTE?

=>yes

"START EXECUTION!"
```

FA	8000
EN	10400
HSS	7000
RE	7200

"CONTINUE SPEAKING ?"

=>yes

"READY TO SPEAK"

=>what scientists came from the USSR ?

"DOES THE FOLLOWING CORRESPOND TO THE INPUT PHRASE ?"

(find "*IIASA-scient" =$nn : c USSR =$c)

=>yes

EXECUTE?

=>yes

"START EXECUTION. IT TAKES TIME"

zimin	USSR
vasiliev	USSR
surguchev	USSR
shigan	USSR
rakhmankulov	USSR
kononov	USSR
klementiev	USSR
kiseleva	USSR
golubev	USSR
dashko	USSR
chebotaryev	USSR
butrimenko	USSR
briabrin	USSR
albegov	USSR

"CONTINUE SPEAKING ?"

=>yes

"READY TO SPEAK"

=>

who came from the USA , their christian names and assignments ?

"DOES THE FOLLOWING CORRESPOND TO THE INPUT PHRASE ?"

(find "*IIASA-scient" =$nn : c USA =$c ; cn =$cn ; a =$a)

=>yes

EXECUTE?

=>yes

"START EXECUTION!"

welsh	USA	william	HSS
schafir	USA	kurt	MT
rogers	USA	andrei	HSS
pahner	USA	philip	HSS
orchard-hays	USA	william	SDS
matthews	USA	william	RE
linnerooth	USA	joanne	SDS
levien	USA	roger	adm
hansen	USA	niles	HSS
gros	USA	jacques	SDS
foell	USA	wesley	RE
dennis	USA	robin	RE
demb	USA	ada	MT
casti	USA	john	SDS
buhring	USA	william	RE
blum	USA	edward	HSS
bell	USA	charles	EN
balinski	USA	michel	SDS

"CONTINUE SPEAKING ?"

=>yes

```
 "READY TO SPEAK"

=>the maximum external budget and the goals of IIASA areas ?

 "DOES THE FOLLOWING CORRESPOND TO THE INPUT PHRASE ?"

 (find *IIASA =$nn : * resarea ; (budg ext) =$ext ; goal =$goal)

=>yes

 EXECUTE?

=>yes

 "START EXECUTION.  YOU RELAX"
```

FA	0	(food rsrcs & tchnlgy, rqrmnts, cnstr, strtgy)
EN	11210	(energy rsrces, demand, optns, cnstr, strateg)
SDS	0	(optimiz, netw, data bases)
MT	165	(plann, mgm, dyn of techn change, inf technol)
HSS	1342	(health care,hum settl,migr,popul growth)
RE	4182	(water dmnd, mgm, transf, hydr models,climate)

```
 (maxim (budg ext) : 11210 EN)

 "CONTINUE SPEAKING ?"

=>no

 "YOU ARE OUT OF SPEAKING !"

%
```

Appendix 2. The protocol examples of DILOS interaction with "Personnel Data Bank"

=>what american or russian scientists work at SDS , their names ?

Balinski	USA	SDS	Michel
Briabrin	USSR	SDS	Victor
Butrimenko	USSR	SDS	Aleksandr
Chebotarev	USSR	SDS	Spartak
Dashko	USSR	SDS	Valeri

=>who of them has a fix-term contract and was appointed before Jan 1976 ?

Balinski	fix-term	(1975 Sep)
Butrimenko	fix-term	(1974 Jan)
Chebotarev	fix-term	(1976 Jan)
Dashko	fix-term	(1974 Aug)

=>who came from France or FRG ?

Beaujean	France
Grenon	France
Hafele	FRG
Raquillet	France

=>when did they receive phd ?

Beaujean	nil
Grenon	nil
Hafele	1955
Raquillet	nil

=>their fields of interest and experience ?

Beaujean	(, math economics)	lect
Grenon	en-resource	(, prof consult "ind-firm")
Hafele	nucl-phys	(, prof consult)
Raquillet	urb-plan	consult

=>what IIASA scientists came from the USA , their names ?

Afifi	USA	A.A.
Balinski	USA	Michel
Bell	USA	Charles
Carter	USA	Harold
Dennis	USA	Robin
Edwards	USA	Ellen
Ferrell	USA	George
Fischer	USA	David
Sherill	USA	Koren

=>who of them was considered by RP before Jan 1977 ?

Balinski	(1977 Jan)
Bell	(1976 Aug)
Edwards	(1976 Jun)
Sherill	(1977 Jan)

=>in what areas they work , their experience ?

Balinski	SDS	(, prof consult)
Bell	ENP	(, consult "ind-firm")
Edwards	CS	comp-prog
Sherill	HSS	nil

=>what scientists from France and Austria were appointed before Dec 1976 ?

Bodenseher	Austria	(1974 Feb)
Breitenecker	Austria	(1976 Oct)
Bruckmann	Austria	(1973 Nov)
Fleissner	Austria	(1975 Jan)
Grumm	Austria	(1975 Sep)
Schlifke	Austria	(1976 Jan)

=>who has a short-term contract , when did they receive phd ?

Briabrin	short-term	1967
Ferrell	short-term	1976
Raquillet	short-term	nil

=>Balinski home institution ?

Balinski	Universite Scientifique et Medicale de Grenoble , France

=>who works with SDS , their expiration terms ?

Balinski	SDS	(1978 Aug)
Bodenseher	SDS	(1977 Jun)
Briabrin	SDS	(1977 Mar)
Butrimenko	SDS	(1977 Dec)
Chebotarev	SDS	(1978 Jun)
Dashko	SDS	(1977 May)
Ferguson	SDS	(1978 Apr)
Grumm	SDS	(1977 Aug)

=>from what cost centers are russians paid , their present grade ?

Briabrin	820	USSR	D1
Butrimenko	820	USSR	S4
Chebotarev	810	USSR	C8
Dashko	(, 810 150)	USSR	D1
Dobrov	770	USSR	F2
Golubev	690	USSR	E1

=>what americans are paid from the same cost centers ?

Balinski	USA	810
Fischer	USA	770

```
=> + (find  =nn  : c France ; ar ; tsk ) ?

Beaujean        ENP    ENP2
Grenon          ENP    (, ENP1 ENP3 ENP7)
Raquillet       HSS    HSS3


=>how many full professors do we have , their nationalities ,
  field of interests ?

Balinski        prof    USA     math
Bruckmann       prof    Austria (, math statistics)
Carter          prof    USA     agricultural economics
Dobrov          prof    USSR    nil
Golubev         prof    USSR    (, hydrol glaciol)
Grenon          prof    France  en-resource
Hafele          prof    FRG     nucl-phys
Majone          prof    Italy   statistics


=>who of them was born before 1934 ?

Balinski        (1933 Oct)
Carter          (1932 Dec)
Dobrov          (1929 Mar)
Grenon          (1928 Sep)
Hafele          (1927 Apr)
Majone          (1932 Mar)


=>Dobrov ?

 (Dobrov
  (* sc
     pv
     (Gennady (1929 Mar)
              USSR
              "res-schol"
              "USSR NMO"
              "Inst Cybernetics Acad Scien Ukranian SSR"
              (, "dep-dir" prof)
              nil
              1950
              1967
              "fix-term"
```

```
              MT
              MT2
              770
              (, "tsk-leader" scient)
              (1976 Oct)
              (1977 Oct)
              nil
              F2
              45474
              F2)))
```

```
=>who from the USA was recommended by himself , their title ,
  experience ?

Afifi         USA    self      res-schol       asso-prof
Carter        USA    self      res-schol       prof
Ferrell       USA    self      guest-schol     consult
Fischer       USA    self      res-schol       (, "asso-prof" consult)
```

```
=>who is paid from 820 cost center  ?

Bodenseher          820
Briabrin            820
Butrimenko          820
```

```
=>what is the institute position of UK scientists ?

Agnew         scient        UK
```

```
=>Interests of Balinski , Butrimenko , Hafele , Majone ?

Balinski      math
Butrimenko    (, math phys "comp-netw")
Hafele        nucl-phys
Majone        statistics
```

```
=> + (time) ?
```

```
tty st user    prio sked  ps   shar+priv    wait    cpu command
3: 0 W victor    10    6  2765  4.6+ 2.9  tty3     29.1 %
5: 0 W victor    40  127  2831  4.6+ 2.9  wait() 128.7 %
5: 0 W victor     1    5  2858       34.0  pipe39 135.2 1110.speak
5: 0 W victor    40    2  2859       32.8  wait()  86.7 1110.find
5: 1 R victor   103    1  2880        8.0             0.7 /bin/ps lc
26 processes; total core load  25.3+133.7= 159 K words
```

=>nil !

"END OF DILOS OPERATION !"

$

INGRES--A Relational Data Base System

M. Stonebraker

In 1970 E.F. Codd [1] proposed the relational model of data and claimed that relational data base management systems had significant advantages over other approaches. They have two basic characteristics.

A Simple Data Model

There is only one data structure (a relation or table) for the user to be concerned with. This is in contrast with hierarchical and network models where the data structures are more complex. This concern for simplicity should be compared with the "structured programming" ideas of Dijkstra and others [2] where the notion of simplicity is advocated for general purpose programming. In both cases the idea is to permit the user (programmer) the least amount of complexity possible so he has a better chance of understanding his problem.

A Powerful Data Manipulation Language in which Storage Details are Absent

The basic notion here is to hide from the user all details of how his data are stored. This allows applications programs to be as simple as possible. Moreover, a powerful data manipulation language greatly reduces the amount of programming effort necessary to implement and maintain an application. Since software costs are rising and hardware costs are falling, this is an especially attractive notion.

Other data manipulation languages are much less powerful and allow the application programmer to manipulate storage details. This second characteristic is a radical departure from other proposals.

The claimed advantages of relational systems were:

- Simplicity of the data model.
- Decrease in the complexity of application programs.
- Since storage details are hidden, they can be adapted to changing user requirements without affecting application programs. Hence, programs can continue to operate over

changes in the way data are stored. This "data independence" is not present in other proposals.

- Protection of data, guaranteeing data integrity for concurrent updates, etc., may be easier.

During the period 1971 to 1974 there was considerable debate concerning Codd's proposal. Some claimed:

(1) A relational data base system could not be implemented efficiently.

(2) Programmers would not be able to understand the proposed relational data manipulation languages. They would be much more comfortable with lower level languages closer to COBOL.

(3) More complicated data models included relations as a special case so why not allow greater flexibility?

(The debate over simplicity is elegantly stated in the early Turing Award lectures. For simplicity E. Dijkstra "The Humble Programmer" (1972) and against simplicity C. Bachman "The Programmer as Navigator" (1973). Both lectures were reprinted in the *Communications of the ACM* (about June 1973 and 1974 respectively).) At the same time, most people questioning the relational model were simultaneously advocating that the 1971 proposal of the Committee on Data System Language (CODASYL) be adopted as a national standard. This proposal (often called the DBTG proposal) had none of the claimed advantages of relational systems. However, by 1973 there was an implementation of the proposal (IDMS).

It was clear to us at Berkeley in 1973 that three steps were required:

- A serious implementation of a relational system to answer point (1).

- Within the implementation context, a design of a "more friendly" data manipulation language to answer point (2).

- If such an implementation were successful and if users could be found to try out the system then they could validate claimed relational advantages and answer point (3).

In late 1973 we embarked on an implementation. No commercial vendors were inclined to try since relational systems were ill understood, speculative, unsure of marketplace acceptance, etc. Hence, we viewed it an appropriate research endeavor. (At about the same time a group at IBM Research also embarked on an implementation. To our knowledge these are the only serious implementation efforts with the two characteristics.)

Our first working prototype was operational in early 1975. Since that time we have extended its features and tuned it to have reasonable performance. Moreover, we have managed to attract about thirty installations to experiment with INGRES or use it in a production mode. Basically we view this implementation effort as having the following two major goals:

- Execution of the three steps mentioned above, and
- Serve as an example to stimulate the commercial marketplace to provide relational systems.

In the process of achieving the first goal we are well along. However, much work still remains to be done. The accepted proposal to ARO indicates the nature of this work and it will not be discussed further here. We expect at the end of the current three year contract period to completely achieve the first goal. Only in the area of geographically distributed data on multiple machines are we uncertain over achieving our implementation goals.

Concerning the second goal there has already been some marketplace response. Systems that look "quasi relational" (e.g. NOMAD by National CSS) and relational systems with a lower level data manipulation language lacking the second characteristic (e.g. MAGNUM by Tymeshare) have very recently surfaced. Implementations by MRI Corp. and Honeywell that are closer to having both characteristics are now under way. We hope that within five years there will be viable commercial systems.

REFERENCES

[1] Codd, E.F., A Relational Model of Data for Large Shared Data Banks, *Commun. ACM*, 13 (1970), 377-387.

[2] Dijkstra, E.N., O.J. Dahl, and C.A.R. Hoare, *Structural Programming*, Academic Press, London-New York, 1972.

NATURAL LANGUAGE AND KNOWLEDGE REPRESENTATION IN DATA BASE

An Overview of PLIDIS
A Problem Solving Information System with German as Query Language*

G.L. Berry-Rogghe and H. Wulz**

BACKGROUND AND APPLICATION OF THE SYSTEM

PLIDIS (Problemlösendes Informationssystem mit Deutsch als Interaktionssprache) is a natural language information system which is being designed in the context of a project on automated language processing at the Institut für deutsche Sprache sponsored by the Ministry for Research and Technology for the years 1976 to 1977. The present project is in many ways an extension of a previous two-year project which achieved the construction of the experimental question-answering system ISLIB (Informationssystem auf linguistischer Basis) (e.g. [7,8]) based on the simulated problem domain of the stock exchange. Within this framework theoretical foundations were investigated and different approaches experimented with. The PLIDIS project differs from its predecessor in its intention to implement an actual system, whereby our emphasis lies on the adaptation of the methods tried out in the pilot study to a real problem domain and on enhancing the problem solving capacities of the system.

The field of application of PLIDIS will be the control of water pollution. A pilot version of the system is being developed in cooperation with the regional "Department of the Environment" at Stuttgart, which supervises industrial wastes dumped into the rivers of Northern Württemberg.

PLIDIS is scheduled to be used in the following capacities:

- As a supervision system, e.g. to check the chemical composition of the samples, to compare the current sample with previous samples from the same firm and to issue appropriate warnings if a norm has been transgressed;

- As an information system, e.g. to answer queries concerning the composition and toxicity of certain chemicals, the

---

*The research reported here is supported by the Federal Republic of Germany's "Bundesminister für Forschung and Technologie" under grant Nr. 081 5900 69 within the "3. DV-Programm der Bundesregierung".

**The authors are indebted to W. Brecht, W. Dilger, R. Guntermann, D. Kolb, M. Kolvenbach, A. Lötscher, H.D. Lutz, K. Saukko, G. Zifonun who collaborate within the PLIDIS project and who did a lot of the research reported here.

characteristics of the production processes of the firms involved, etc.;

- As an investigation system, e.g. to detect where pollution may have originated and possibly suggest plans of actions to be taken.

GENERAL DESIGN OF PLIDIS

The PLIDIS information system is composed of a linguistic-logical part, which translates the German input into an internal representation modeled on the predicate calculus, and a problem solving part, which, in addition to performing the usual storage and retrieval functions, involves problem domain-specific regularities in the deduction process.

The design of the system is largely modular and allows extensive user interaction between the various execution phases. This modularity is an essential prerequisite for efficient teamwork as each member of the group can be allocated a specific part and possible changes in personnel take place smoothly. Interactive facilities are essential to facilitate experimentation and debugging.

Figure 1 is a diagrammatic representation of the system's main components showing the flow of information between them.

The PLIDIS user has several choices of access to the system, some of which are designed especially for a more naive user and some destined for the system designer and administrator. The natural language processor (NLP) enables the user to formulate problem descriptions as natural language questions or to use natural language for the input of shorter pieces of information such as rules about his problem domain or data for updating. For the input of stereotyped data of larger quantities, the user may have data sheets on his terminal, which are processed by the processor of formatted input (FIP). This processor also provides facilities accessible by the system's command language (CL) to define new data sheets and procedures for plausibility checks of the formatted input. The NLP and the FIP have the same task to perform, i.e. to translate the input into the language of internal representation (IR), an extension of first order predicate calculus. The processor for information and problem description (PIP) either stores the incoming information or activates problem solving mechanisms in the case of problem descriptions, according to the type of question asked.

In the current state of the system, the processor for answer-formulations (PAF) generates only some sort of "pretty-print" from the formulas of internal representation that contain the information found by the PIP component as answer to the users' questions. It would be desirable at a future stage that this component be replaced by procedures that generate natural language

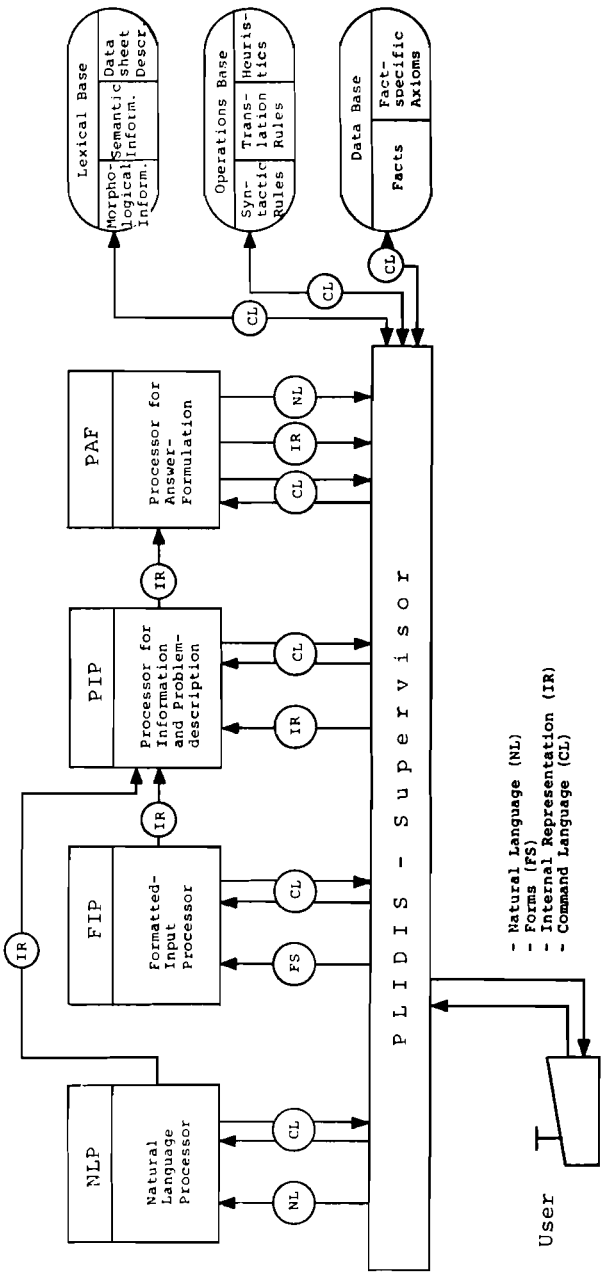


Figure 1. PLIDIS--main components and information flow.

sentences out of IR formulas. The interaction of these components is guided by the PLIDIS supervisor which processes the command language statements and accepts also INTERLISP code. The command language gives the nonnaive user access to various interactive facilities that are helpful for testing and debugging. The algorithms draw on lexical and operational information contained in external data bases supplied by the user/designer:

- The morphosyntactic lexicon contains at the moment some 10,000 entries of nonlemmatized word forms with their morphosyntactic features such as tense, number, gender, etc.

- The semantic lexicon contains information about a word's equivalent in the internal representation such as relational symbol, operational symbol, individual term, its "sort" (see next section), the number and sort of the arguments for each predicate, and so on.

- The data sheet inventory contains the various data sheets for entering mass data such as laboratory reports, particulars about the firms, etc.

- The syntactic rules specify a grammar for German as an augmented transition network (ATN).

- The translation rules specify "transformations" of the parsings of the NL sentences into the internal representation.

- Heuristics specify syntactic and semantic criteria to guide the problem solver.

The data base proper or the "knowledge" of the system is a collection of atomic formulas in the internal representation stating the following information about the problem domain: *mass data* about samples of river water, the legal norms of the allowed concentrations, the composition of various chemicals, their toxicity, etc., information about the firms being controlled (type of plant, production processes, treatment of waste, etc.), *axioms* stating general logical implications as well as specific regularities in the world model such as "x is greater than y implies that x is not equal to y" and "if a chemical interferes with the river flora, it is toxic".

THE INTERNAL REPRESENTATION IN A SYMBOLIC LANGUAGE (KS)*

General Considerations

The choice of an appropriate IR for the knowledge within the PLIDIS system was not motivated solely by theoretical considerations but by its use for effective retrieval of answers to queries

*Konstruktsprache in German.

stated in natural language. An IR for a Question-Answering(QA) system must have the following properties:

- Expressive power to match the complexity of natural language;
- World-modeling capacity to describe all situations, events, actions, and changes of states occurring in a given microworld; and
- Deductive capacity pertaining to the solution of problems put to the system.

The broad aspects can be made more explicit in the following specific requirements.

(I) Like NL, the IR must be an "object language", i.e. it should not describe regularities of the German language, but should act on the same referential level as NL. This entails that it should not contain metalinguistic symbols such as set-theoretic ones, cases, etc.

(II) The IR should be able to describe arbitrary microworlds, i.e. for any given concrete microworld, it should have the means to designate all typical entities existing in that world: individuals, sets of individuals, events, processes, actions, etc. Similarly, it should be able to express time, temporal relations, and causality.

(III) The syntax of the IR must be explicitly described in a grammar. This grammar guides automatic mapping processes of NL structures into IR structures and allows the problem solver to operate on the syntactic level of the IR.

(IV) With the IR must be associated a formal semantic interpretation that accounts for the way in which IR formulas correspond to particular arrangements in the external world and furthermore allows one to decide about the equivalence of formalisms [6].

(V) It should be suited to the application of general formal deduction mechanisms, so that it is not necessary to program specific deduction algorithms for each deduction (in the sense of "methods")--which does not of course exclude the use of heuristics.

In particular, points (IV) and (V) indicate the use of a predicate calculus (PC) for the internal representation, as PC is interpreted by a formal semantics in the form of Tarskian model theory and a general "theorem prover" mechanism operates on it.

The standard first-order PC does not, however, fulfill all the above requirements (e.g. condition (V)). Therefore a symbolic language (KS) was designed modeled on the first-order but incorporating a number of extensions. In the discussion below, we distinguish between the formal representation language KS, which according to requirement (II) is independent of the given

microworld, and concrete KS languages defined by a world-specific vocabulary. The general construction rules of the IR language KS are described next; a preliminary outline of the concrete language KS water pollution control is given later.

Short Description of the Syntax of KS

In addition to the usual sets of symbols in a PC--namely, predicate symbols, individual symbols, connectives, and quantifiers--the vocabulary of KS contains the set S of "sorts":

$$S = \{uni, obj, int, sit, per, ort, zus, akt, \ldots\} \quad .$$

(These names are abbreviations for the German: Universal, Objekt, Intervall, Situation, Person/Personenkörperschaft, Ort, Zustand, Aktion.) The set SV of sort-indexed variables is the Cartesian product of the set $V = \{x_1,\ldots,x_n\}$ of variables and the set S of sorts. KS terms can be constructed with the aid of operation and relation symbols. For each such symbol, the sorts of its arguments are specified. The sort of the term thus constructed is determined by the sort of the last argument of the operation or relation symbol. The following conditions of well-formedness for terms are defined:

(1) Sort-indexed variables from SV are terms.

(2) Individual constants are terms. To each constant is assigned a member of the set S.

(3) Let F be an n-place operation symbol, to which is assigned an n+1-tuple of sorts:

$$\langle a_1,\ldots,a_n,a_{n+1}\rangle \quad (a_i \in S) \quad .$$

Let $t_1^{a_1},\ldots,t_n^{a_n}$ be terms of the sorts $a_1,\ldots,a_n$ respectively. Then

$$\left(F\ t_1^{a_1},\ldots,t_n^{a_n}\right)$$

is a term of the sort a_{n+1}. Operational terms are in general individual terms. If the n-th argument term $(t_n^{a_n})$ is of the sort "int" (interval), then the term designates individuals with reference to a particular time. Such individuals are states of the world, actions,

processes, and so on. They are terms of the sort "sit" (situation) and are made up of an operation symbol followed by the following tuple of sorts:

$$\langle a_1,\ldots,a_{n-1},\text{int},\text{sit}\rangle \quad .$$

(4) Let R be an m-place relation symbol, to which is assigned an m-tuple of sorts:

$$\langle a_1,\ldots,a_m\rangle \quad (a_i \in S) \quad .$$

Let $t_1^{a_1},\ldots,t_{m-1}^{a_{m-1}}$ be terms of the sort $a_1,\ldots,a_{m-1}$.

Then $\left(R\ t_1^{a_1},\ldots,t_{m-1}^{a_{m-1}}\right)$ is a term of the sort a_m. Relational terms are "list terms". Such terms designate sets of individuals (see later).

Atomic formulas in KS are constructed according to the following conditions of well-formedness:

(5) Let F be an n-place operation symbol with the tuple of sorts

$$\langle a_1,\ldots,a_n,a_{n+1}\rangle \quad .$$

Let $t_1^{a_1},\ldots,t_n^{a_n},t_{n+1}^{a_{n+1}}$ be terms of the sorts $a_1,\ldots,a_{n+1}$. Then

$$\left(F\ t_1^{a_1},\ldots,t_n^{a_n};t_{n+1}^{a_{n+1}}\right)$$

is an operational atomic formula.

(6) Let R be an m-place relation symbol with the tuple of sorts

$$\langle a_1,\ldots,a_m\rangle \quad .$$

Let $t_1^{a_1},\ldots,t_m^{a_m}$ be terms of the sorts $a_1,\ldots,a_m$. Then

$$\left(R\ t_1^{a_1},\ldots,t_m^{a_m}\right)$$

is a relational atomic formula.

Nonatomic formulas are constructed according to the usual construction rules of PC. A more detailed description of the syntax of KS can be found in [15,16].

Special Features of KS

Many-Sortedness

The set S of sorts can be extended as demanded by the requirements of specific fields of application. The sorts underlie a hierarchical structure made use of in problem solving. The sortal structure of KS imposes semantically motivated conditions of syntactic well-formedness--in the sense of Katz-Fodor "selection restrictions". But the specification of the number and sorts of the arguments of a predicate is made as a function of the world-model, rather than being guided by linguistic principles.

The advantages of a sortal structure in a representation language were indicated in [5]. A logical sortal calculus with linguistic considerations was proposed in [11].

Complex Term Building

The notion of "term" in KS is defined recursively, so that it is possible to embed terms within terms, thus reflecting more closely some NL constructs, such as complex noun groups.

Example: "the mother of the neighbor of the friend of Hans" becomes in KS: (MOTHER (NEIGHBOR (FRIEND HANS)))
In the framework of the concrete KS-language with reference to social relations, MOTHER would have been defined as a 1-place operation symbol taking the tuple of sorts ⟨per, per⟩. FRIEND and NEIGHBOR would have been defined as 2-place relation symbols also with the tuple ⟨per, per⟩.

Quantification

In KS the NL quantification symbols VIELE, MANCHE, EINIGE (many, several, some) are defined. They describe the size of sets of entities. The same applies to the natural numbers which can

also be used as quantification symbols. They underlie the following conditions of well-formedness:

Let QU be a quantification symbol and let $\left(R\ t_1^{a_1},\ldots,t_{m-1}^{m-1}\right)$ be a list term of the sort a_m, then $\left(QU\ \left(R\ t_1^{a_1},\ldots,t_{m-1}^{a_{m-1}}\right)\right)$ is a quantified list term of the sort a_m.

Plurality

Singular and plural objects can be designated in KS by "individual terms" and "list terms" respectively. As an example the KS representation of the sentences "der Nachbar der Mutter von Hans ist Fritz" and "die Nachbarn der Freunde von Hans sind Franz und Egon" is given:

```
(NACHBAR (MUTTER HANS); FRITZ)
(NACHBAR (FREUND HANS); (LISTE FRANZ EGON))
```

Arithmetic Operations

KS incorporates arithmetic operations such as PLUS, DIFFERENCE, TIMES... which can be interpreted as LISP functions.

The KS Language for the Control of Water Pollution

In PLIDIS a concrete KS language is defined that derives its vocabulary from the field of application in the control of water pollution. Some examples of the vocabulary of KS water pollution control are given below:

- Individual constants: ARSEN (sort : "stoff")
  ZYANID (sort : "stoff")

- Operation symbols:

  PROBE (2-place; sortal tuple : ⟨betrieb, int, stoffkoll⟩)
  PROBENEHMER (1-place; sortal tuple : ⟨stoffkoll, per⟩)
  ANTEIL (3-place; sortal tuple : ⟨stoff, stoffkoll, physobj, num⟩)
  LABORBERICHT (3-place; sortal tuple : ⟨perkörp, stoffkoll, int, physobj⟩)
  BETRIEB (2-place; sortal tuple : ⟨firma, ort, betrieb⟩)

- Relational symbol:

  GIFTIG (1-place; sortal tuple : ⟨stoff⟩)

The above predicates can be "translated" into English as follows:

> PROBE = "sample", PROBENEHMER = "sampler", ANTEIL = "amount", LABORBERICHT = "laboratory report", BETRIEB = "firm", GIFTIG = "toxic".

The following is an example of a KS-term:

> PROBE (BETRIEB MAX-MÜLLER STUTTGART) 76.Ø1.13.14.ØØ)
> "The sample taken from the firm Max Müller in Stuttgart on 13.1.1976 at 14.00 hours."

The following is an example of a KS formula:

```
(ANTEIL ZYANID (PROBE (BETRIEB MAX-MÜLLER STUTTGART) 76.Ø1.13.)
        (LABORBERICHT (BETRIEB CHEM-UNTERSUCHUNGSANSTALT PLOCHINGEN)
                      (PROBE (BETRIEB MAX-MÜLLER STUTTGART) 76.Ø1.13.)
                      76.Ø1.15.)
        ;(Ø,5 mg/l))
```

"The amount of cyanide contained in the sample taken from the firm Max Müller in Stuttgart on 13.1.76, according to the laboratory report of the chemical analysis center in Plochingen produced on 15.1.76 amounted to 0.5 milligram per liter."

NATURAL LANGUAGE ANALYSIS IN PLIDIS

The requirement of modularity in a system such as PLIDIS is dictated not only by organizational reasons, but also, from a more systematic point of view, it was desirable to maintain a strict separation between components that are theoretically or methodologically well understood or are of no central interest to the project, and those that are topics of genuine effort and experimentation and where research is still going on.

Thus the natural language processor is separated into three passes (see Figure 2): a PASSØ for the morphological identification, a PASS1 for syntactic analysis and a PASS2 for code generation, i.e. translation into the language of internal representation.

PASSØ: Morphological Identification

At an early developmental stage of the system PASSØ was a program for morphological analysis that operated with a lemmatized dictionary. For each German word of the system's vocabulary there existed only one dictionary entry, the basic form of the word. A certain class of verb forms, for example, were represented by their infinitive form. It was the task of the program to apply morphological rules to inflected forms of words and to reduce them to their basic form, which then allowed a dictionary look-up for further information. This analysis was very time

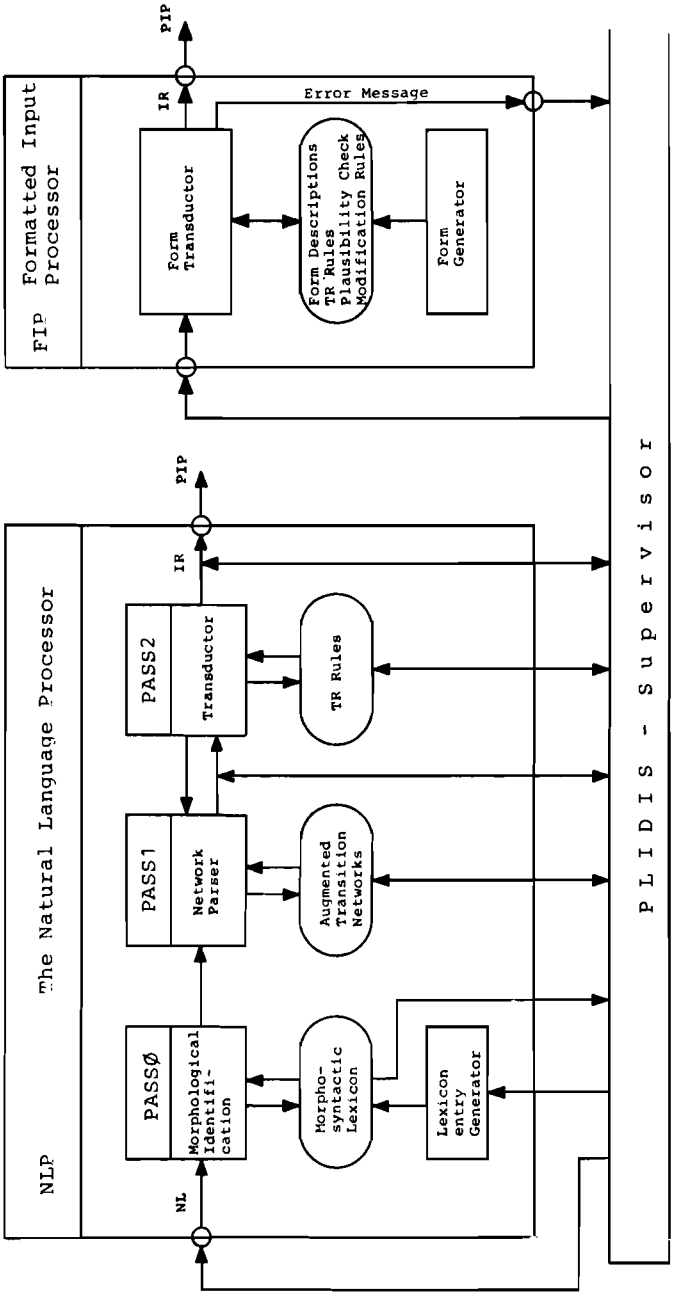


Figure 2. PLIDIS components for processing natural language input and input by data sheets.

consuming and was replaced by a very simple program that works with a nonlemmatized dictionary. Each inflected form of a word has an entry in the dictionary with its full morphological information such as basic form of the inflected word, word class, gender, tense, etc. The dictionary is stored on an external device with index sequential access such that the time required for morphological identification of a word is rather small. The amount of work necessary for entering all inflected forms of a word into the dictionary is reduced by a special function of PASSØ, which generates from a dictionary entry of a basic form the entries for the inflected form. Finally a HELP routine of PASSØ enables even a user with little linguistic knowledge to write the dictionary entry for a basic word form.

PASS1: Morphosyntactic Analysis

The parsing of the German input sentence is done by means of an ATN as described in [13]. This technique was chosen as it seemed to be the best studied of parsing techniques and at the same time it may be handled easily by linguists without any special training in programming. The advantage of the ATN language is its open-endedness, allowing the definition of new arcs, tests, and actions as required for the analysis of specific NLs. In the present version the parser is able to recognize the majority of German sentence structures, including complex sentences containing all types of subordinate clauses--relative, adverbial, object and subject, etc. Complex noun phrases having inflected participal constructions as attributes such as "Das von der Firma Müller in den Rhein eingeleitete Abwasser" can be handled. The verb phrase may contain a main verb in any tense or mode, with the exclusion of the conjunctive mood.

Because of their inherent ambiguity, some constructions not resolvable by purely syntactic criteria had to be excluded:

- Coordination between noun phrases (e.g. "Die alten Männer und Frauen");
- "Elliptical" noun phrases, i.e. noun phrases without a nominal head (e.g. "Er nannte das billigste gut").

The ATN for German is very weakly structured, in particular with regard to the noun phrase. Whereas a linguist would like to have a structure something like Figure 3, PASS1 produces an analysis as shown in Figure 4 for the sentence: "Der Anteil an Zyanid in der Probe der Firma Müller betrug 2 mg/l." (The amount of cyanide contained in the sample of the firm Müller was 2 mg/l.) Certainly it would be possible to push the noun phrase analysis further by extending the syntactic categories and by using information such as dependency frames of verbs. But since deeper noun phrase analysis sooner or later needs semantic information, it was decided to restrict the syntactic analysis to the generation of a list of the main constituents of the input

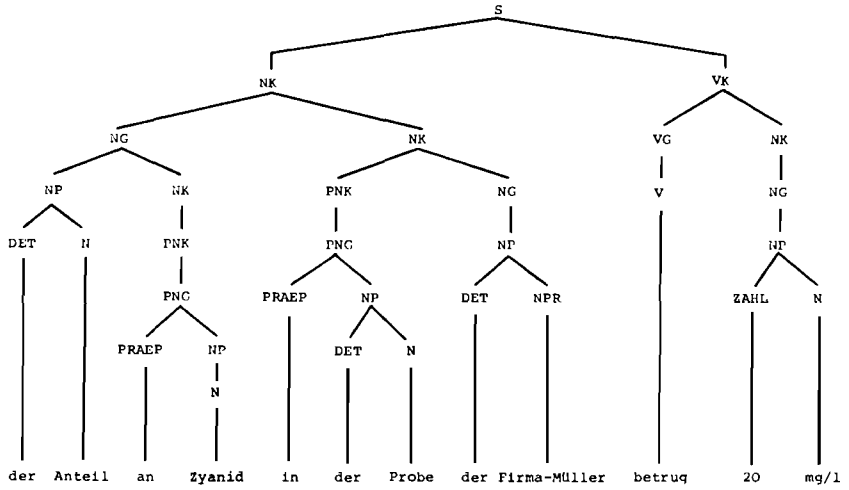


Figure 3. Example for desirable syntactic structuring within the domain of noun groups.

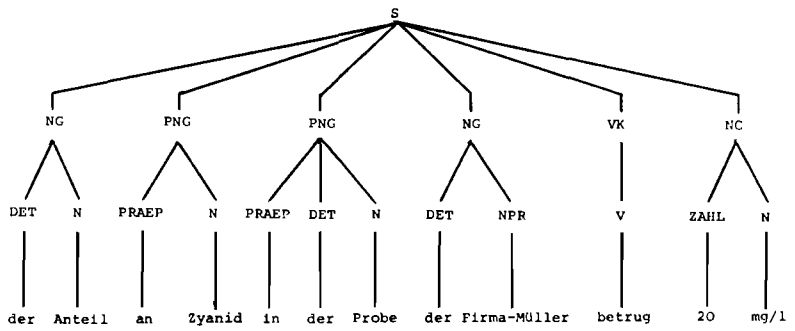


Figure 4. Example for structuring capacity of the PLIDIS NL parser.

sentence with a minimal dependency structure and to pass the burden of semantic interpretation to the translation component in PASS2.

PASS2: Semantic Analysis Component

Within the PLIDIS system semantic analysis is viewed as the problem of translating NL sentences into formulas of the internal representation language KS; more precisely, to generate KS code from the parsing trees produced by the network parser of PASS1. In the earlier ISLIB approach augmented transition networks were used to state the rules for KS code generation. As stated earlier, this approach turned out not to be very efficient and remained at an ad hoc level, since it was not possible to find a theoretical foundation that would have allowed reduction of the number of rules needed with this approach. The new concept for the NL to KS translation starts from the concept of a translation grammar for two languages L_1, L_2, where L_1 is the source language and L_2 the goal language of the translation [14]. PASS2 then can be viewed as a program that interprets the translation grammar rules. The translation grammar may be compared with a transformational grammar [4], the rules of which operate on already existing derivation trees of a phrase structure grammar of the source language, i.e. German in the PLIDIS system. The nodes of these trees are labeled with nonterminal (syntactic categories of the grammar) and terminal symbols (source language words) of the phrase structure grammar. In a similar way the translation grammar rules are applied to the derivation trees of the source language, which correspond within the context of PLIDIS to the lists of bracketed and labeled constituents from the parsing of NL sentences.

For the sake of simplicity and clarity, the translation grammar is explained here by simplified examples and in an abbreviated terminology of derivation trees. The translation grammar consists of three types of rules:

- Rules for the replacement of source language symbols--i.e. in general NL words--by the context pattern of their goal language equivalent;
- Insertion rules for the goal language context pattern; and
- Pattern raising rules.

The rules are based on the concept that it will be possible to define for each goal language symbol something that we will call a context pattern. The context pattern of a symbol is a prediction about the syntactic context in which this symbol will occur. Thus the writer of a translation grammar for German to KS may state in a rule of the first type that the KS symbol PROBE

may correspond to the German word "Probe" (sample). The grammar of KS defines that PROBE may be used within a two-place ⟨TERM⟩ of the sort ⟨stoffkoll⟩, where the first argument has to be a ⟨TERM⟩ of the sort ⟨firma⟩ and the second argument a ⟨TERM⟩ of the sort ⟨int⟩. Thus in any context where the German "Probe" is translated by the KS symbol PROBE, it will be followed by two terms specified as above and the context pattern for PROBE can be defined as a tree structure, where the top node is labeled by ⟨TERM ; stoffkoll⟩ and the terminal nodes by (, PROBE, ⟨TERM ; firma⟩, ⟨TERM ; int⟩ and) respectively. The rule of the first type for the German word "Probe" would state, then, that "Probe" is to be replaced by the described context pattern (see Figure 5).

In the context pattern of PROBE, ⟨TERM ; stoffkoll⟩ is viewed as the head of the context pattern, whereas the nonterminal KS symbols ⟨TERM ; firma⟩ and ⟨TERM ; int⟩ are considered as "slots" and it is the task of the second type of rule of the translation grammar to define how to fill in these slots. A distributional analysis of the context of German "Probe" will show that the nominal attributes of "Probe", i.e. a noun group in the genitive case or a prepositional noun group following "Probe", are the constituents the translations of which have to be inserted into the slots of the PROBE context pattern. Thus the insertion rules for the KS context pattern assigned within the translation of a natural language sentence to German "Probe" would state that the slot with the name ⟨TERM ; int⟩ has to be filled with a context pattern of the same name, resulting from the translation of a prepositional noun group following "Probe", specifying also possible prepositions like "am" or "vom".

Example:

Let $RR_1,\ldots,RR_6$ denote some rules of the first type for the replacement of German words by the context pattern of their KS equivalent,

IR_1, IR_2 rules of the second type for the insertion into context patterns;

let $\in$ denote the empty context pattern, consisting of no symbols.

The application of these rules to German "Probe" within the context "die Probe bei Müller & Co vom 15.12.76" (the sample from Müller & Co of 12/15/76) can be represented schematically as shown in Figure 5, where the arcs stand for the application of the rules that label the arc.

The use of the sorts of KS for disambiguation within the translation can be shown if one considers "die Probe *von* Müller & Co *am* 15.12.76" as an alternative formulation for "die Probe *bei* Müller & Co *vom* 15.12.76". As the insertion rule IR_2 requires the translation of a prepositional noun group with the preposition "von" or "am" to be inserted as tense argument into

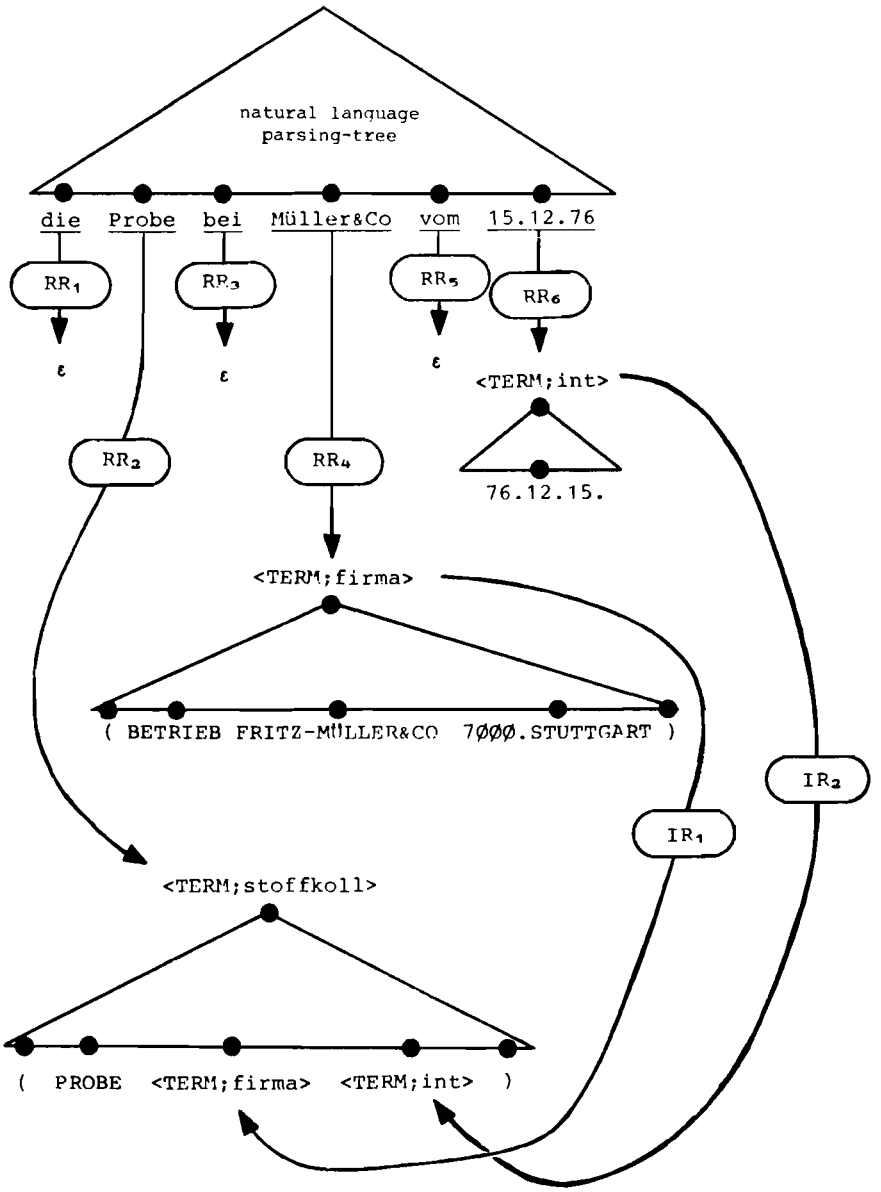


Figure 5. Simplified illustration of the application of replacement rules (RR) and insertion rules (IR).

the context pattern assigned to "Probe", the translation of "Müller & Co" would take the place of the second TERM within the PROBE pattern. But since the KS equivalent to "Müller & Co" is a TERM of the sort ⟨firma⟩, a check of the sort consistency will block the insertion at the TERM place with the sort ⟨intervall . For each insertion rule there is a side effect defined. If a filled-in context pattern is inserted into the slot of another pattern, it is deleted at its original place, i.e. replaced by the empty pattern ∈ (see Figure 6).

If all terminal symbols, i.e. all NL words of a derivation tree, are replaced by the context pattern of their KS equivalent and if all slots of these patterns are filled in, the pattern raising rules may be applied to the remaining structure in the following ways:

(1) A nonterminal symbol x of the source language grammar can be replaced by a filled-in context pattern if this pattern is dominated by x and if all other context patterns dominated by x are equal to the empty context pattern.

(2) If a nonterminal symbol x of the source language grammar dominates only empty patterns, then it is replaced by the empty pattern.

(3) If the top node of the remaining tree structure is labeled by a symbol of the goal language grammar, a head y of a context pattern can be replaced by the string that results from the concatenation of the symbols dominated by the head y under the condition that y does not dominate another head of a context pattern.

For simplicity we will illustrate the application of the pattern raising rules with an abstract example.

Example:

Let A, B, C, D be some nonterminal symbols of a source language grammar and a, b, c, d, e, f symbols of the goal language grammar; let PR_1, PR_2, PR_3 denote the pattern-raising rules as described above in (1), (2), (3) respectively. Figure 7 then illustrates the application of these rules to the tree whose top is labeled by A and where a and d are the heads of context patterns. The numbers preceding the rule names indicate the order in which these rules were applied. If the string resulting from the application of the pattern raising rules consists of terminal symbols of the goal language grammar, then a translation has been found.

Since various details of a translation grammar for a subset of German into KS are still subject of experimentation the PASS2 program, which interprets the translation rules, has not yet reached its definitive form.

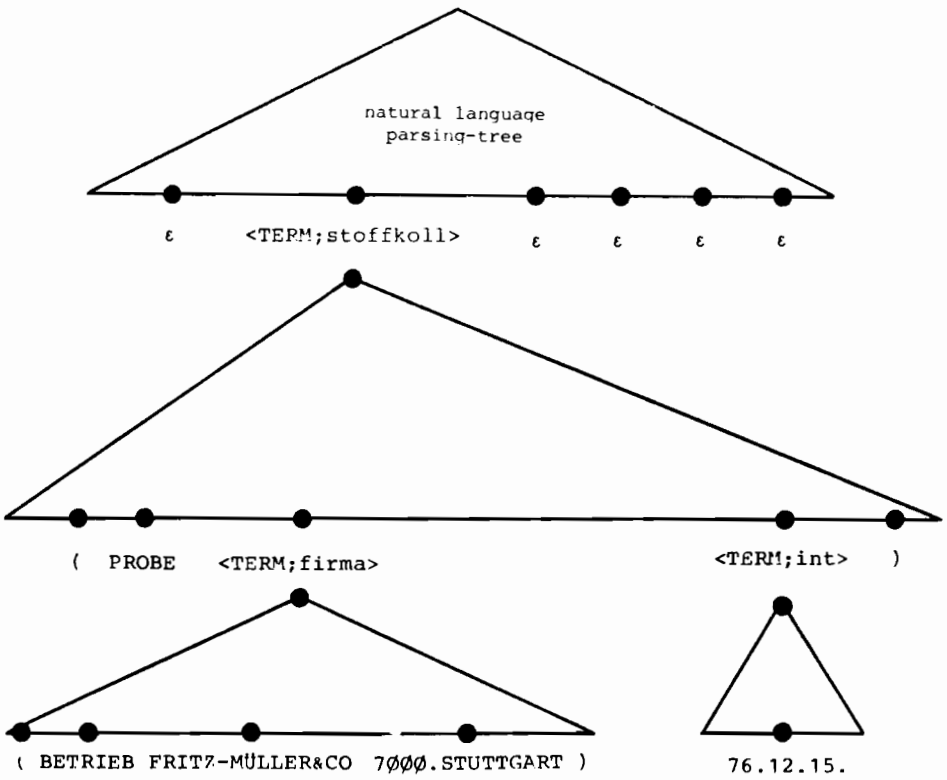


Figure 6. Result of the application of the rules of type (1) and (2) on "die Probe bei Müller & Co vom 15.12.76".

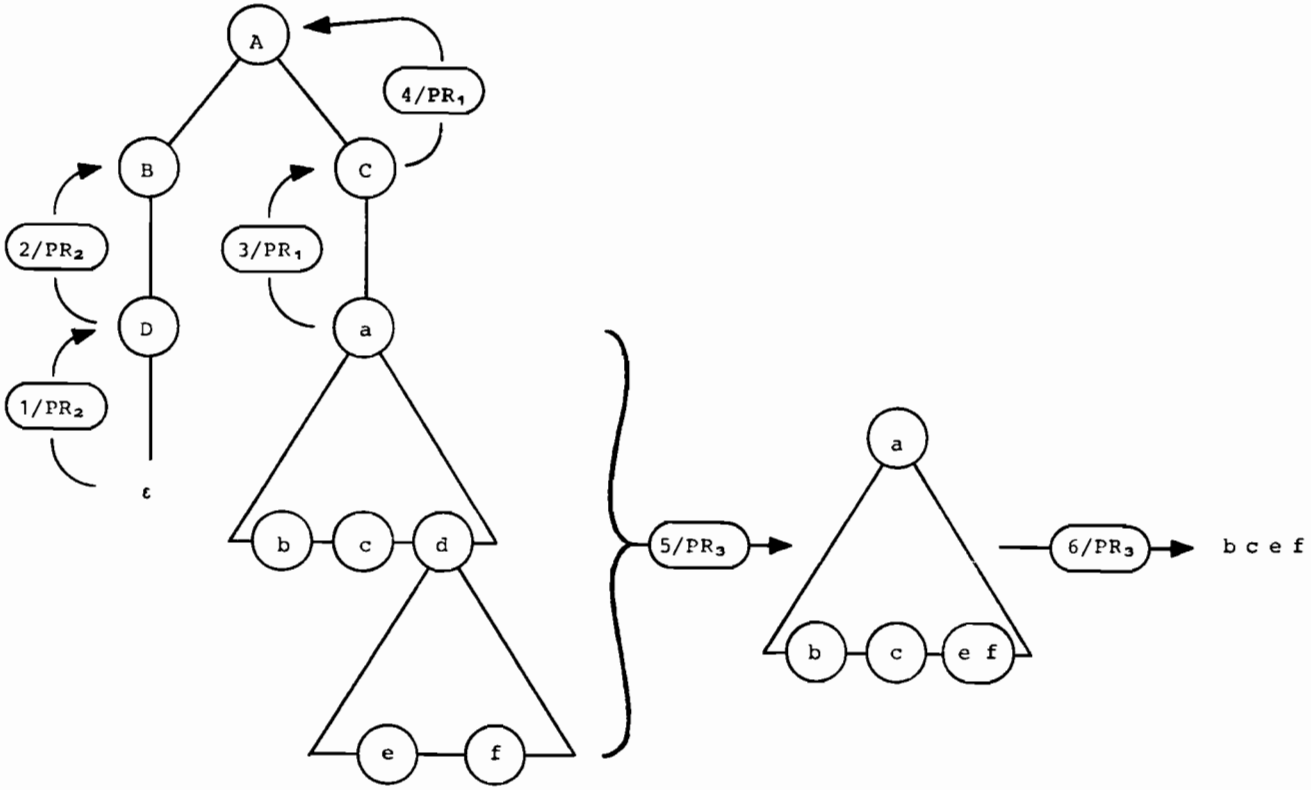


Figure 7. Application of pattern raising rules (PR).

INFORMATION HANDLING AND PROBLEM SOLVING

The processor for information and problem description consists of data base management procedures for storing the symbolic data into the data base, and "problem solving procedures" for the answering of questions. This section deals primarily with the latter, as data base management problems are not within the main topics of the PLIDIS development and will arise only in the "real-life" application of the system, when mass data have to be processed. When the PLIDIS data base management is too weak to handle these problems, the component may be replaced by adaptations of already existing data base management systems.

The problem solving component of PLIDIS must be able to perform the following operations:

- matching,
- set-theoretic,
- deduction, and
- arithmetic.

The choice of the appropriate matching operations depends mainly on the techniques used in the "data base management" component for storage of mass data (hash-coding, pattern matching, etc.).

As NL questions put to the system usually involve the use of plural noun phrases, it should be possible to ask for the extension of sets (of individuals or of mass-terms). This task is performed by a component called "Terminterpreter" (TI), which reformulates the KS question into set-theoretic terms and subsequently evaluates this term with set-theoretic operators. The deduction process proper is done by means of a theorem prover (TP) based on the resolution principle.

Arithmetic operations present no particular problem, as they are represented as KS operators evaluated as LISP functions. The components performing the above operations interact with each other in the process of solving a particular problem. This interaction is guided by a "monitor".

The theorem prover based on the resolution principle proceeds in two main stages: normalization and resolution. The process of normalizing consists in reducing the KS formulas into sets of literals obtained out of clauses in conjunctive normal form, the existential quantifiers having been replaced by skolem constants or functions. For greater efficiency, normalizing takes place when the formulas are entered into the data base, so that it only needs to be carried out once. Questions must of course still be normalized by the TP. The process of resolution proper generally involves two important aspects: search strategies and

heuristics. Under the heading search strategies fall such alternative techniques as "state space" versus "problem reduction", "depth first" versus "breadth first" analysis, and the use of connection graphs as described in [9] supported by methods such as the Waltz algorithm. Each of these techniques presents advantages for particular types of problems. It was deemed important in the PLIDIS implementation of the TP to allow the deduction strategies to be kept *variable*, according to the type of problem at hand. In a QA system, for example, the "problem reduction" method (such as "input resolution") has the advantage that the question being asked (i.e. the conclusion) can be taken as the starting clause, thus ensuring that only clauses containing a predicate relevant to the question are resolved upon. Because of the incompleteness of input resolution conclusions from false premises ("ex falso quodlibet") are avoided. On the other hand, where the TP is to be used for controlling pollution, the goal state is in general not known and a "state space" deduction method is hence indicated.

The "default" implementation of the PLIDIS theorem prover functions in state space mode with breadth first analysis. It is possible to change the operation mode to either "unit resolution" or "input resolution", by specifying the appropriate parameters. The axioms being resolved upon are linked by a connection graph [cf.3]. It is envisaged that connection graphs will be constructed when the data are entered in the data base. The set of clauses can thus be divided into subsets linked by a connection graph, representing different miniworld models of related axioms. As a further extension, heuristics could be similarly connected into subsets, which would aid the selection function. Whether the entire system's knowledge can thus be neatly divided into subsets has not yet been empirically verified. On a preliminary investigation, it seems that at least certain coherent bodies of knowledge can be distinguished, such as legal norms, geographical data, composition of chemicals, etc.

At each step in the deduction process, the selection of the next pair of clauses to be resolved upon is guided by a "selection function". This function calls upon semantic as well as syntactic heuristics. An example of a syntactic heuristic would be a function computing the size of the unifier, i.e. the number of substitutions. (For example, if the unifier of the link k contains p elements then f(k) = 1/p; another example would be the use of resolution with unit clauses--the value of this function would be either '0' or '1'.) Semantic heuristics take into account the semantic characterization of the predicate and the arguments of the literal. Such heuristics must be formulated in terms of the world model and the problem at hand.

Finally, the PLIDIS problem solver makes use of the sortal structure of KS in selecting a unifier for a set of clauses. Before a substitution is carried out, it is checked if the sort of the constant is compatible with the sort of the argument, as

illustrated by the following two clauses:

$$\neg(\text{AT } x\ y) \vee \neg(\text{MOVE } x\ y\ z) \vee (\text{AT } x\ z)$$
$$(\text{AT table } (\text{PLACE table}))$$

The following unifier can be established together with a specification of the sortal characterization of the substitutions

(table PHYSOBJ)/x,((PLACE table)LOC)/y
(table PHYSOBJ)/x,((PLACE table)LOC)/y

yielding the following resolvent:

$$\neg(\text{MOVE table } (\text{PLACE table}) \vee (\text{AT table } z)$$

The above clause is ill formed, as the first argument of MOVE has to be of the sort "animate"; the substitution must hence be rejected.

IMPLEMENTATION OF PLIDIS

PLIDIS is written in SIEMENS-INTERLISP, which is an implementation of Uppsala-INTERLISP [12], on a SIEMENS-4004/151 running under the BS 2000 operating system. Uppsala-INTERLISP is itself an implementation of INTERLISP [10] for an IBM 360/370 configuration. No specific SIEMENS-INTERLISP features were used so that the system will almost certainly run in other INTERLISP implementations.

REFERENCES

[1] Chang, C.L., and R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1970.

[2] Dilger, W., Ein Frage-Antwort-System auf der Basis einer prädikatenlogischen Sprache, in *Proceedings of the Workshop on "Dialoge in natürlicher Sprache und Darstellung von Wissen"*, Freudenstadt, 1976.

[3] Dilger, W., *Verbindungsgraph und Auswahlfunktion*, internal paper, Institut für deutsche Sprache, Mannheim, 1976.

[4] Ginsburg, S., and B. Partee., A Mathematical Model of Transformational Grammars, *Information and Control*, 15 (1969), 297-334.

[5] Hayes, P.J., A Logic of Actions, in B. Meltzer and D. Michie eds., *Machine Intelligence*, No. 6, University Press, Edinburgh, 1971.

[6] Hayes, P.J., Some Problems and Non-Problems in Representation Theory, in *Proceedings of the 1974 AISB Summer Conference*.

[7] Kolb, D., and H.D. Lutz, *Verarbeitung von Netzwerken*, ISLIB-Info I-4, Institut für deutsche Sprache, Mannheim, 1975.

[8] Kolb, D., and H. Wulz, *Allgemeine Beschreibung und Kurzanleitung für die Benutzung von ISLIB Börse*, ISLIB-Info I-1, Institut für deutsche Sprache, Mannheim, 1975.

[9] Kowalski, R., A Proof Procedure Using Connection Graphs, *Journal of the ACM*, 22 4 (1975).

[10] Teitelman, W., *INTERLISP Reference Manual*, XEROX Palo Alto Research Center, Palo Alto, California, 1974.

[11] Thomason, R., A Semantic Theory of Sortal Incorrectness, *Journal of Philosophical Logic*, 1 (1972), 209-258.

[12] Urmi, J., *INTERLISP /360 and /370 User Reference Manual*, Uppsala University Data Center, Uppsala, 1975.

[13] Woods, W.A., An Experimental Parsing System for Transition Network Grammars, in R. Rustin, ed., *Natural Language Processing*, Algorithmics Press, New York, New York, 1973.

[14] Wulz, H., *Konzept einer Theorie einer Übersetzungsgrammatik*, unpublished manuscript, Institut für deutsche Sprache, Mannheim, 1976.

[15] Zifonun, G., *KS: eine formale Sprache zur kanonischen Darstellung natürlicher Inhalte in einem automatischen Frage-Antwort-System*, internal paper LDV-MA-73-3, Institut für deutsche Sprache, Mannheim, 1974.

[16] Zifonun, G., *Die Konstruktsprache KS, Entwurf eines Darstellungsmittels für natürlichsprachlich formulierte Information*, internal paper, Institut für deutsche Sprache, Mannheim, 1976.

An Overview of OWL, A Language for Knowledge Representation*

P. Szolovits, L.B. Hawkinson, and W.A. Martin

OVERVIEW AND MOTIVATION

We have undertaken the design and implementation of a new computer language for knowledge representation, called OWL. We have become convinced that recent progress in linguistics and in artifical intelligence (AI) now suggests a set of principles worth implementing as part of a programming language to make them uniformly accessible for our further work.

For a computer program--as for a person--it is more effective to know how to do something than to be able to figure it out. The AI field has made important progress under an opposite set of assumptions: that all knowledge of the domain should be expressed in propositional form and that a program's actions should be directed by a general purpose problem solver operating on propositions representing the application world. Such a problem solver would always figure out what to do next based on the state of the world and its set of general principles. At the same time, most programs that have been used for their ability to perform in an application domain rather than for their pedagogic clarity have used a very different form of organization: the knowledge of how to perform the task was implicitly built into the steps of the program. Of course, such an organization is generally accompanied by inflexibility, difficulty of extension, incomprehensibility and unprovability of the program, and many other ills. If, however, we could express the description of the procedural knowledge of the program in the same formalism as its declarative knowledge of the domain of application, then both would be equally accessible. This is precisely what is done in OWL--the program is just another aspect of the description of the application world, and knowledge of how to solve specific problems of that world can be explicitly embedded in the description.

We have taken English as the basis for our knowledge representation formalism. The greatest attraction of this approach is that it almost trivially satisfies our need for expressive power. After all, native speakers of English can usually communicate

---

*This work was supported by the Advanced Research Projects Agency of the US Department of Defense and was monitored by the Office of Naval Research under contract #N00014-75-C-0661.

their knowledge of any domain of interest in English,* perhaps augmented by specialized notations and vocabularies particular to the domain. Because we choose a computer representation designed to be similar to the natural language (NL) employed by a computer-naive user of one of our programs, we expect that the translation process from English sentences to our internal structures will be straightforward. Once we succeed in translating the English phrase into our internal representation, that will allow *all* of OWL's activities, including understanding the sentence in semantic detail, resolving references, mapping the sentence onto some capability of the system for acquiring new knowledge or answering on the basis of old, etc., to make use of the same representational formalism. This, in turn, will help us to make the complete operation of the program accessible for explanation to, and modification by, someone who may well understand the domain of application but not our computer technology.

Arguments for English as a programming language have been made since the early 1960s, yet it has not been universally acclaimed as desirable. The principal objections to basing a programming language on English (or any NL) center on the innate ambiguity of NL and its lack of conciseness when contrasted with special mathematical notations. The second problem is rapidly resolved if we extend our definition of NL to allow the incorporation of new notations. After all, the NL of a physics text is hardly the literary English of the day. The first problem has both a trivial and a difficult component: pure syntactic ambiguity, as created by the existence of homonyms for instance, is simply controllable, whereas ambiguity arising from the fact that what one (literally) says is not what one actually means is, of course, difficult. Our response is simply that we wish to begin by representing precisely what one says, and we will allow the determination of the meaning of each utterance to be part of the problem that the system is to solve.

During the past few years, we have implemented the following components of a complete system based generally on the above ideas:

- A Linguistic Memory System (LMS) [3], which is a memory (data base) of concepts in which all knowledge in OWL resides. LMS can be viewed as a semantic network, with a somewhat unusual interpretation of its nodes and arcs.

---

*We limit ourselves to "left-hemisphere knowledge", which does not include visual skills or manipulative skills where local muscle/nerve training is an essential component. Thus, our domains are restricted to reasoning tasks where the necessary data about a problem can be acquired verbally, e.g. medical diagnosis and treatment of the type that could be done by consultation over the telephone (probably not, for example, diagnosis of skin disease, where visual inspection is a critical skill), automatic program writing, question answering.

- A theory of English grammar which specifies how any utterance of English can be represented in terms of LMS concepts.

- A skeletal world model organized as a taxonomy of concepts and intimately related to the theory of English grammar.

- An augmented transition network parser to translate English utterances into their OWL representations.

- A generator to perform the inverse transformation to the parser.

- An interpreter which carries out procedures represented in the OWL formalism.

- An explainer which provides English explanations (via the generator) of procedures and data dependencies known to the interpreter, as well as results of previous executions of those procedures.

These components are at differing stages of development. We are pursuing a breadth-first approach to implementation, where we try to have some version of each of these components before trying to have the "ultimately" correct version of any of them.

In terms of the above components, we have been building the following programs:

- Programwriter, which takes a declarative specification of simple programs that need to be written and designs, optimizes, and codes them. The scope of its capabilities includes programs to maintain bank balances and sell tickets for scheduled events [5].

- Susie Software, which is another automatic programmer, for writing manipulation programs for the block's world. It is a research environment for developing a discourse model that lets Susie engage the user in a dialogue concerning the program it is trying to write [2].

- Proctor, which helps a business manager to design a procurement system. It is an "unstructured" questionnaire that provides a framework for a manager to think about his system requirements [1].

- A Digitalis Therapy Advisor, which makes clinical judgements about the condition of a patient who is receiving the drug digitalis, makes further therapeutic recommendations, and can interactively explain its reasoning steps to the user [8].

- A question-answering system for a relatively simple data base.

We will give an overview of LMS, the theory of grammar, and the interpreter, and discuss other modules as they relate to those central components.

THE LINGUISTIC MEMORY SYSTEM

The OWL LMS is a semantic network with a single primary data type, the concept, and a secondary data type, the symbol. Symbols are merely strings of characters that denote senses of English words and affixes and have no innate significance. Concepts represent the meanings of all words, phrases, clauses, sentences, etc., of English as well as any needed nonlinguistic entities. It is very important to note that, whereas in a traditional semantic network each node of the network represents a single word or item, in LMS each node represents any of the higher level constructions mentioned above. Thus, where a typical semantic net would identify the meaning of a sentence as some subnet of the whole network, LMS identifies it as a single node of the network.

The Essential Structure of Concepts

Concepts, the nodes of LMS, have structure. In fact, we will concentrate on the essential structure of a concept as the primary organizational facility of LMS.

Every concept is defined by a pair, (genus specializer), the essence of that concept. The genus is another concept, and the specializer is either a concept or a symbol. The genus specifies the general type of the concept; if the genus of concept C is B (i.e., if C = (B specializer)), then we imply that C is-a B, or C is a kind of B.* C is called a specialization of B, and B is called a generalization of C. The specializer serves to distinguish this concept from all other concepts with the same genus; it does not by itself define the concept.** The genus and the specializer together identify a concept.

We want to interpret all the concepts in LMS as forming a single taxonomy or tree-like classification system in which the genus points "up" in the taxonomy. To do so, we must designate a single concept, SUMMUM-GENUS, whose genus is itself. That condition makes SUMMUM-GENUS the root of the tree. Further, we insist that no loops may occur in the expression of concepts in terms of themselves or each other (with the above exception for SUMMUM-GENUS). Then, all concepts will form a tree structured

---

*The general implication of is-a or is a kind of (AKO) links is that "something" (properties, features, place of classification, ways to treat, etc.) is inherited by C from B. We will define this more precisely later.

**For example, we may represent "dog house" as (HOUSE DOG) and "dog tail" as (TAIL DOG), and although both concepts are specialized by DOG, they are clearly different.

classification: starting from any concept in the conceptual memory and successively moving to its genus will always lead to the root concept SUMMUM-GENUS in a finite number of steps. That number will be called the genus depth of the concept. We also introduce a notational convenience. So far, we have only allowed a concept to be written as (genus specializer). But clearly, the depth of parenthesization for writing any concept will be at least its genus depth, and this is terribly inconvenient. Thus, we allow equivalence declarations, such as A = (B C), which allows any appearance of A to stand for an appearance of (B C). A is called the label of (B C).

The notion of derivative subclassification [3] complicates this picure somewhat. It assures that all specializations of a concept are classified the same way the specializers themselves are classified in the conceptual memory. For example, if in the taxonomy both DOG and PIG have genus ANIMAL, then we classify (TAIL DOG) and (TAIL PIG) under (TAIL ANIMAL). The generalizer of a concept (A B) is the most specific specialization of A whose specializer is a generalization of B, or, if there are none of these, just A itself.* The genus of a concept is thus always either its generalizer or the generalizer of its generalizer, etc. By moving along the successive generalizers from any concept, we must finally reach SUMMUM-GENUS, and the number of steps required is called the generalizer depth of the concept.

We have now described some of the essential structure of each concept, thus each node, of a conceptual memory. Before we turn to arguing for the utility of this structure to represent knowledge, let us see what the essential structure of the nodes already implies for the semantic network as a whole. In our current implementation, every concept is directly linked to its generalizer and specializer. Every concept is not, however, linked directly to its genus, since the genus can easily be computed from generalizer and specializer links. A typical, but very small, conceptual memory taxonomy is shown in Figure 1.

Attachment

In the previous section, we presented the essential structure of a concept in LMS. The act of creating a new node in LMS is called specialization, and we say that we specialize a genus, G, by a specializer, S, to form the concept (G S). As we shall

---

*An intermediate concept in the taxonomy, such as (TAIL ANIMAL) in our example, is automatically created by LMS whenever more than one concept may be classified under it. Thus, the generalizer of a concept, and hence the number of times that we need to move from a concept to its generalizer in order to reach its genus, will depend dynamically on what other concepts are in the taxonomy.

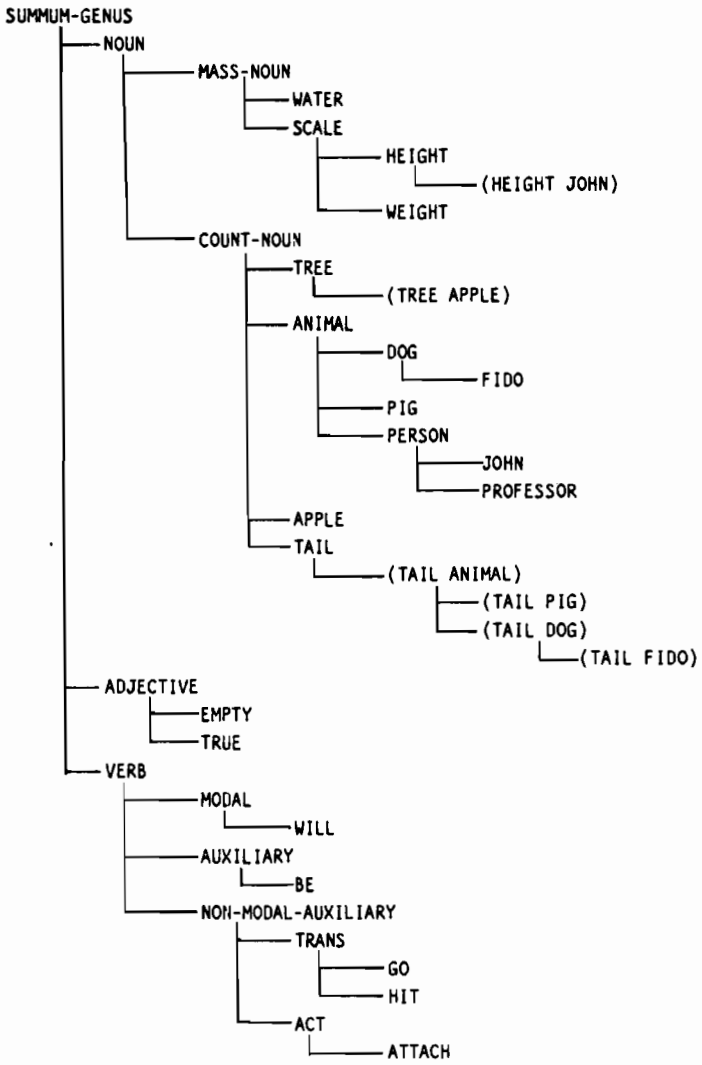


Figure 1. A sample conceptual memory taxonomy. This shows the classification of some of the concepts used in this paper into a small conceptual memory taxonomy. The taxonomy is a tree which is shown in the figure by successively indenting branches, as in an outline. Note that derivative subclassification causes the subtree under TAIL to be organized in a similar way to the subtree under COUNT-NOUN. This sample is of course very small and sparse; the taxonomy we currently use has nearly three thousand concepts and a correspondingly more complex organization.

argue, any phrase of English can be suitably encoded as a single concept (though of course it may be a very complex one). When we wish to reason with a concept, however, we will find it convenient to introduce an epistemologically distinct meta-level representation. For example, if the concept C encodes the sentence "John Smith is a good man" and we wish to represent our belief that C is true, we cannot merely encode with D that "That John Smith is a good man is true", because now the question of D's truth is open for discussion.* We retreat to a formal meta-level to make statements about elements of our universe of discourse which are to be taken at face value rather than be subject to interpretation. With such an ability for meta-level description, we see that if C is marked as TRUE at the meta-level, then that is a stronger statement than D. From the former, the Interpreter may conclude C's truth absolutely, while from the latter, only conditionally on D's truth.

The act of attachment creates a directed link in LMS between two nodes. We write [A B] and say that B is attached to A. Unlike specialization, attachment creates no new concepts. It merely establishes an (unlabeled) link from A to B. The meaning of the connection will depend completely on what A and B are and on whatever is interpreting the connection. We give a few illustrative examples of attachment here:

- All concepts B whose generalizers are the concept of A are automatically attached to A and are called its indexed branches because they are classified directly under A in the specialization taxonomy.

- Some concepts (C A) are attached to the concept A and are called its indexed aspects. For example, (AGE JOHN) may be attached to JOHN and encodes JOHN's AGE aspect.

Note that both of the above forms of attachment are easily recognizable because the concept to which attachment is made appears as the generalizer or specializer, respectively, of the attached concept. They derive from the essential structure of concepts and serve much the same purpose for the conceptual memory as do index entries in a book's index. These attachments do not really

---

*It is not merely the representation of truth that is at issue here. A similar treatment is necessary for supposition, hypothesis, "possible futures", and in fact all the fundamental knowledge on the basis of which OWL operates. Of course the effect of the meta-level statements that we allow could alternatively be introduced by suitable conventions for the Interpreter. For example, we could adopt the convention that any statement about which no qualifying information is known is true. We prefer, however, to make such a convention part of the Interpreter and not part of the semantics of LMS.

bear information; they are established when the taxonomy is built and are not subject to interpretation or change. Thus, the use of attachment, a meta-level operation, is appropriate.

- Values may be specified by attachment: e.g., [(AGE JOHN) 49].
- Attributes or descriptors may also be attached: e.g., [JOHN MIDDLE-AGED], [(AGE JOHN)(EQUAL (TO (AGE MARY)))].
- Characterizations may also be specified by attachment: e.g., [JOHN PROFESSOR]

This second set exemplifies storage of information (facts in the object domain), yet we are representing such information at the meta level. This is because we intend that reasoning be based on these facts without further verification. We are willing to guarantee their truth in this domain of application.

HOW ENGLISH PHRASES ARE REPRESENTED AS CONCEPTS

In this section, we shall first argue informally that the combination of concepts through specialization provides a mechanism capable of representing English phrases. We will then extend our notion of specialization to deal more rigorously with some problems we encounter.

What Does an English Expression Say?

We view English phrases as expressions built up by combination. To explore what forms of combination are necessary, we examine some modes of communication in English and see how they are achieved by combining words and phrases.

Designating

We use a conventional name for a concept the listener may be assumed to know. In its simplest form, the conventional name is a word of English, e.g., "apple", which we represent in OWL by APPLE = (FRUIT "APPLE").* But we need many more conventional names than we have words in our language. So, we permit the formation of conventional names as combinations (pairs). One member of the pair indicates the class of the concept, the other provides a distinguishing, or specializing, element to make the pair unique. For example, "apple tree" is a conventional name formed by specialization. In LMS, we represent it as (TREE APPLE). Note that no strong distinction is made between conventional names that are compound words and those that are phrases in English. Compare "fire hydrant", (HYDRANT FIRE), and "fireman" (MAN FIRE).

---

*"APPLE" is the LMS notation for the symbol "apple". The concept (FRUIT "APPLE") is LMS's notation for the English concept apple.

Identifying

We identify an unnamed concept by combining its class and some (restricting) modifiers. For example, "tall tree", (TREE TALL), and "the apple tree in my yard", (((TREE APPLE) THE) (IN (YARD MY))).* The difference between identifying and designating is often slight. In designating, we assume that the hearer already knows the concept, whereas in identifying, we ask him to come to know it from what he knows of its components and whatever else we may later tell him. Thus, a "shoe tree", which we might initially accept as an identifying compound without a conventional designation, may come to designate a concept if shoe trees become a popular consumer item. Just as compound words develop from conventional names that are phrases, the latter develop from identifying phrases.

Specifying a Grammatical or Interpretive Aspect

Chiefly by word affixes, English marks phrases and gives clues to their use in forming sentences and to their proper interpretation. For example, for "books" (BOOK -S), the -S is a grammatical marking for plural on the base concept BOOK. In "hitting" (HIT -ING), and "to jump" (JUMP TO), the -ING and TO play a similar role. This form of marking is called inflection. In LMS, inflection is expressed by specializing the concept to be inflected by the affix (or other marker).

Specifying a Semantic Aspect

We also represent semantic aspects by specialization. For example, "size of apple" (SIZE APPLE).

Predication

When we want to say something about an object or action in a factual or hypothetical context, we use predication. Jespersen [4] calls this nexus:

> If we compare *the red door* and *the barking dog*, on the one hand (junction) and on the other *the door is red* and *the dog barks* or *the dog is barking* (nexus), we find that the former kind is more rigid or stiff, and the latter more pliable; There is, as it were, more life

*Some linguists might feel that this phrase should have a different structure, such as (((TREE APPLE)(IN (YARD MY))) THE). We do not claim to have the final answer to all such structural questions, but our formalism allows us to capitalize on whatever insights linguists may have. Where structures are in dispute, we have chosen what seems best to us.

> in it. A junction is like a picture, a nexus is like a drama or process. In a nexus something new is added to the conception contained in the primary: the difference between that and a junction is seen clearly by comparing, e.g.
>
> The blue dress is the oldest.
> The oldest dress is blue.
> A dancing woman charms.
> A charming woman dances.

In our terms, a junction identifies or designates. A nexus, or predication, makes a statement and depends on interpretation for its meaning.

In LMS, we introduce a new notation to express predication: subject/predicate. For example, Jespersen's sentence "the oldest dress is blue" becomes ((DRESS OLDEST) THE)/BLUE. For uniformity of representation and implementational convenience, however, we will implement predication in LMS using specialization by adopting the following convention: The predication A/B will be implemented as ((B NEXUS.) A).

Itemization

To specify a group of things related in some simple way, we itemize them. Particular types of itemization are: sequences, conjunctions, disjunctions, sums, products, contrasting pairs, etc. For example "red, white, and blue", "3+5+9", and "input/output" are all itemizations. LMS introduces an external notation for such itemizations but implements them by a conventional use of specialization and attachment. The details are unimportant and will not be pursued here. We should add, however, that we feel the notion of sequence to be fundamental.

Naming

This important mechanism of English will play a major role in our representation formalism. Language often uses context to say concisely what might otherwise require a verbose specification. In particular, we often use part of a compound to name the whole: GENERAL for (OFFICER GENERAL), CAPITAL for (LETTER CAPITAL), and EMPTY for (CONTAINER EMPTY). In each of these cases, the specializer in context names the whole concept. We shall encounter more general uses of naming below.

Kinds of Specialization

Our treatment of specialization as outlined above is inadequate for some subtler issues of representation. Although we

have identified several uses of compound formation in English communication, we have represented them all by the same specialization operation. We form, in a completely similar manner, compound phrases like "the dog" (DOG THE), "sheep dog" (DOG SHEEP), "small dog" (DOG SMALL), and "dog in the yard" (DOG (IN (YARD THE))). For these examples, no problems arise because we can recapture from the specializer itself the kind of compound we have formed. But that will not generally be the case, as we shall see below. In this section, we introduce seven distinct kinds of specialization to enrich our representation scheme.

The English phrase "fat man" is ambiguous. In its common meaning, it stands for a man who is overweight to some degree. The same phrase, however, also describes a professional circus performer of great girth, with whom we associate characteristic forms of dress, behavior, etc. In terms of the modes of communication listed above, we are either designating the circus performer by his conventional name or identifying the man who is overweight by his genus and a distinguishing characteristic.* OWL is unique in that we make a procedural distinction between these two senses of "fat man". In the first case, "fat" is combined with "man" to identify a pattern in memory, and then that pattern is used to find the referent. In the second case, "man" alone is used to find a pattern in memory, and then items which match this pattern are further checked to see if they pass the pattern designated by "fat". We could imagine a skinny fat man only in the first sense, as referring to the circus performer. But our representational scheme, as presented so far, offers only (MAN FAT) for "fat man", and fails to distinguish the two senses we have discussed.

To preserve the desired distinction between these readings of "fat man", we will mark every specialization with its meta type, which indicates the relation between the concept and its genus.** We will represent our overweight man by a restrictive specialization, (MAN*R FAT). A restriction (A*R B) may always be paraphrased as "an A which is B", e.g. "a man who is fat", and a restriction always represents a concept which is a kind of its genus with the additional attribute which is its specializer. Note that a tall fat man ((MAN*R FAT)*R TALL) is not the same as a fat tall man ((MAN*R TALL)*R FAT), either in real life or in conceptual memory. In a stereotype (A*T B) the specializer has

---

*In spoken language, the compound representing the conventional name is spoken almost as if it were the compound word "fatman". This additional clue is not available to us via written language.

**We are introducing a minor inconsistency here, because we change the meaning of "genus" somewhat. By the rules of LMS, the genus of the concept (A*R B) is A*R, yet we will refer here to A, the concept's linguistic genus, as its genus.

some close relation to the genus but is not necessarily a property of it. Consider not just our circus performer, (MAN*T FAT), but also (HYDRANT*T FIRE), where the relation between "fire" and "hydrant" is a complex one: "a hydrant which is a source of water with which one can put out a fire".

The seven OWL meta-types and their notational suffixes are:

*R	restriction	*A	aspect
*T	stereotype	*X	inflection
*S	species	*P	partitive
*I	instance		

(A*S B) represents a subspecies of A, where B is often just a symbol. This represents a Linnaean classification system in which we assume that different subspecies of A form mutually exclusive categories. This is a powerful tool for data base search. (A*I B) represents an instance of A. Instances, as species, are mutually exclusive.* We thus provide a distinction between classes and individuals by distinguishing instances from species.

An aspect specialization (C*A B) is a kind of its genus C, that is closely associated with its specializer B. For example, "height of John" (HEIGHT*A JOHN) and "John's leg" (LEG*A JOHN). Aspects also play the traditional role of programming language variables. For example, if we have a recipe for pancakes that calls for one egg, that egg will be represented by (EGG*A (RECIPE*T PANCAKE)).

An inflection (A*X B) is used to specify a grammatical or interpretive aspect. It has the unusual behavior that it inherits properties not only from its genus, as all other specialization types do, but also from its specializer. In fact, properties inherited from the specializer override any inherited from the genus. For example, "books" (BOOK*X -S), is plural even though BOOK is singular, because -S carried the plural property.

The partitive (A*P B) is like a semantic version of inflection. The partitive inherits properties from both its genus and specializer, where context determines the appropriate interpretation. Thus, one may first open and then eat a can of beans, first opening the can and then eating the beans.

The above is a short sketch of our approach to representation. A much more complete treatment will be found in [7].

---

*Some systems further divide instances into manifestations: e.g. "the young Churchill". We would handle this as (CHURCHILL*R YOUNG), where CHURCHILL = (MAN*I "CHURCHILL").

Parsing

To translate from strings of English words to their representation, we use an augmented transition network parser based on [10]. The OWL parser uses no registers but maintains a constituent stack of concepts with each phrase for which a transition network (TN) is being followed. On every arc is an OWL concept which must be matched for that transition to apply and a set of combining functions which manipulate the matching concept and constituent stack.

It is the task of the combining functions to compose OWL concepts representing parts of a phrase into the concept representing the whole phrase. The role of the TN is to invoke the combining functions in the appropriate sequence. The parser operates nondeterministically (via backtracking). Failure leading to backup may occur either because the input string fails to meet word-order constraints (i.e. no match can be found for any arc from a nonterminal node of a TN) or because a combining function rejects a proposed phrase. The conceptual memory contains (expressed via attachment) strictly enforced constraints on case slots of all grammatical concepts. Using these constraints, the combining functions control all compositions such as adjectival and adverbial modification and case assignment for verb phrases. The word-order constraints of the TNs plus the concept-formation constraints in the conceptual memory (as they are used by the combining functions) thus express our grammar.

Two mechanisms of special interest should be mentioned: the use of naming to postpone the introduction of ambiguity, and bidding. Because many English words and phrases have alternate interpretations in LMS (e.g. our "fat man"), if we were to split our computation nondeterministically every time alternative interpretations of a phrase were available, we would spend a lot of processing effort carrying all those interpretations along until all but one could be eliminated. Further, if more than one interpretation succeeded and the sentence parsed ambiguously, we would have a difficult task localizing the cause of the ambiguity. To avoid these problems, we take a "wait and see" approach [cf. 12,6] and try not to choose the appropriate interpretation until some further constraint forces that choice. Postponing the choice is accomplished by use of the naming mechanism introduced above. In our "fat man" example, we say that conventionally we will form the restriction (MAN*R FAT) as the interpretation of the phrase and we will have in the knowledge base an indication that (MAN*R FAT) names (MAN*T FAT). In this case, the distinction may never have to be drawn during parsing, since no grammatical decisions will depend on it, and it will be some later step of reasoning in the system that may have to choose the "circus performer" interpretation.

In a typical situation where grammatical distinctions arise early in parsing, we take a slightly different approach from the previous example. The word "drinks" is either the plural of the

noun "drink", as in "We had a few drinks", or the third person singular of the verb "drink", as in "Joe drinks beer at dinnertime". Here, rather than choosing one of these as a primary interpretation, we create the neutral (DRINK*X -S) and say that it names both (DRINK*X PLURAL-NOUN) and (DRINK*X THIRD-PERSON-SINGULAR-VERB). To make this scheme work, every combining function must succeed not only when the concepts given to it may be directly combined but also when any concepts named by the given ones may be combined. Matching of concepts on TN arcs is similarly augmented. Further, rules like the above for "drink" generalize, and OWL encodes those generalizations rather than specific naming rules for each concept. (These naming generalizations are called productive naming rules. They are applied by the normal inheritance mechanism of LMS, so of course they may be overridden by more specific information in any particular case.)

Bidding is another mechanism for deferring a choice among alternative and avoiding undue nondeterminism. Its application is best seen when considering the attachment of prepositional phrases. For example, in "I rode along the highway in my limousine", we may eliminate "the highway in my limousine" as implausible and attach the prepositional phrase to the predicate (or predication). By contrast, in "I liked the phone in my limousine", the prepositional phrase clearly belongs with "phone". We cannot always make such a definitive judgment: "I saw the man beside our house" places either me or the man beside the house. From further context, the ambiguity may be resolved: "As I approached, I saw the man beside our house". We treat this problem by suspending a path in parsing at a point where it is about to take an arc transition for a prepositional phrase until all possible paths leading to taking such a transition for that same phrase are identified. Then, a conflict-resolving routine is called to permit any number of the possible interpretations to proceed. That routine will, in general, invoke the Interpreter to try to decide which interpretation(s) are best. Its success will depend on the sophistication of world knowledge in the conceptual memory and on the existence of appropriate strategies available to the Interpreter to apply that knowledge. A more specific mechanism which similarly addresses the problem of "selective modifier placement" is presented in [11]. We have not yet made any significant use of this bidding strategy.

REASONING

We have implemented an initial version of an Interpreter for OWL, which is the basis of the system's ability to reason. It is a large program with many interesting capabilities, of which we will here describe only the central ones. Sunguroff [9] describes the implementation details of the current version, Brown [2] is concerned with the use of the Interpreter for dialogue and the handling of failure, Long [5] gives another view of the Interpreter's use for automatic programming, and Swartout [8] discusses the Interpreter's record-keeping and updating capabilities and their relation to explaining program behavior.

So far, we have interpreted OWL concepts as static entities, mere translations of English phrases. The system's action when given the sentence "Prescribe an appropriate dosage of digitalis for Mr. Jones" cannot be merely to translate that sentence into its internal representation and then stop. But how is it to know what the procedural meaning of some sentence is?

If an OWL concept has a METHOD aspect, then it is called a PLAN and is something the Interpreter can carry out. When the Interpreter is called (its argument is the call), it performs the following steps:

- It tries to match the call to a known plan in the knowledge base. The search for a matching plan proceeds "upward" from the call, so that the most specific plan that matches will be selected.*

- It checks that any required properties on the cases (variables) of the plan occur also on the concepts which will be matched to them.

- It creates a new event, to record the initiation of execution of the selected plan, and binds all the matched variables.

- If the plan contains a PREREQUISITE aspect, it checks if it is already TRUE and if not, then it tries to make it true. This subgoal step of course once again uses the Interpreter.

- It carries out the steps of the METHOD, either in parallel or in sequence, whichever is specified.

We attempt always to use the Interpreter to solve subproblems of an initial problem so that the general matching and reasoning resources we build up will be available at all levels. For example, if X is a prerequisite which is not yet satisfied, we merely call the Interpreter with the call (GET*T X). Classical goal-directed behavior can be achieved by use of the PRINCIPAL-RESULT case on a plan, which identifies the teleological goal of the plan. Then, if a GET is unable to find a plan by its upward search of

---

*This is a very important idea. With it, we can embed completely specific plans to solve any problems we know will arise often and will be critical to the system's performance. We also use it to express plans when their choice is dictated not by a reasoned choice but by convention in the application area. If a specific plan is unavailable, slightly more general plans will be attempted, and only if all such plans are found inapplicable will the system resort to some general deductive scheme. We have noted that only when a great majority of specific plans for a domain is available will the system's performance be at an "expert" level. This agrees with our observations that human experts seem to have large portions of their ordinary professional behavior "precompiled" into fixed routines.

the concept tree, it may search for a matching principal result and select the plan that promises that result. One other important aspect of the Interpreter is that after every step of interpretation, it dispatches to its next step through the main top-level loop. There, failure-handling and advice-giving procedures may always be invoked to redirect the course of computation by "backing off" from unproductive lines (if they can be recognized).

We are continuing to refine our understanding of the representation of English phrases in the formal notation of OWL and the use of a complex Interpreter that works within that formalism to perform all reasoning tasks that arise in language processing and various application areas.

REFERENCES

[1] Bosyj, M., *A Program for the Design of Procurement Systems*, TR-160, MIT Laboratory for Computer Science, Cambridge, Mass., 1976.

[2] Brown, G.P., *A System to Process Dialogue: A Progress Report*, TM-79, MIT Laboratory for Computer Science, Cambridge, Mass., 1977.

[3] Hawkinson, L.B., *The Representation of Concepts in OWL*, presented at the 4th IJCAI, Tbilisi, August 1974.

[4] Jespersen, O., *Essentials of English Grammar*, University of Alabama Press, University, Alabama, 1964.

[5] Long, W.J., *A Program Writer*, Ph. D. Dissertation, MIT, Cambridge, Mass. (in preparation).

[6] Marcus, M., Diagnosis as a Notion of Grammar, in Shank and Nash-Webber, eds., *Preprints from the Workshop in Theoretical Issues in Natural Language Understanding*, 1975.

[7] Martin, W.A., *A Theory of English Grammar* (in preparation).

[8] Swartout, W.R., *A Digitalis Therapy Advisor with Explanations*, TR-176, MIT Laboratory for Computer Science, Cambridge, Mass., 1977.

[9] Sunguroff, A., *OWL Interpreter Reference Manual*, MIT Laboratory for Computer Science, Automatic Programming Group, Cambridge, Mass., 1976 (unpublished internal documentation).

[10] Woods, W.A., Transition Network Grammars for Natural Language Analysis, *Comm. ACM*, 13, 10 (1970).

[11] Woods, W.A., An Experimental Parsing System for Transition Network Grammars, in R. Rustin, ed., *Natural Language Processing*, Algorithmics Press, New York, 1973.

[12] Waltz, D.L., *Generating Semantic Descriptions from Drawings of Scenes with Shadows*, TR-271, MIT Artificial Intelligence Laboratory, Cambridge, Mass., 1972.

Progress in the Development of a Multipurpose German Language Question Answering System

E. Lehmann

INTRODUCTION

In recent years major artificial intelligence (AI) research efforts in natural language understanding (NLU) were oriented toward a more sophisticated and generalized knowledge representation [2,7,10,13,15,16,17] and the important role higher level knowledge (frames, schemes, scripts) plays in language processing ([1] as well as Minsky, Abelson, Schank, Rieger, Charniak). Dealing with the intricacies of understanding connected speech [9,18] strengthened the need to organize the interaction and cooperation between many different processes and knowledge sources in an efficient and transparent way.

One of the lessons learned from this effort was that immense masses of knowledge of all kinds are required in a system with generalized capabilities for understanding unrestricted NL. It would be a formidable task to prepare a complete, working system with all the required knowledge and to put it into the computer in the form of program statements, logical axioms, special grammar rules, and linguistic characterizations of lexical entries (as was done in some small-scale AI experiments).

Limitations with regard to speed and working storage and the lack of an adequately developed linguistic theory and of a formalized corpus of knowledge are the main obstacles for building a system with general NLU capability today. Therefore, the only way out for building working NLU systems for special applications now seems to be to impose strong restrictions on the universe of discourse [18,19], the complexity of relevant knowledge, the vocabulary, and the admissible verbal constructions. If we regard the complexity of knowledge and of language structure, the power of inferential capabilities, the depth of conceptual representation, the extent of the dictionary and of the general knowledge available, the changeability of application areas, and the failure-tolerance of the system as independent dimensions of design, then more freedom in one or two of these dimensions has to be paid for by more restrictions in the remaining ones. Consequently, the distribution of freedom or restrictions over these dimensions is of prime importance for designing and judging NLU systems.

The design of most recent question-answering systems (QASs) or NL data base interfaces was in a rigid way oriented toward only one singular application area [19]. The universe of discourse of such systems is often so radically restricted (with

regard to its structure and extent) that the desired advantages of intelligent NL interaction with stored, comprehensive knowledge bases more or less vanish. Also more theoretically oriented approaches for exploring fairly sophisticated mechanisms of NLU (including frames and more human-like inferential processes) mostly led--despite their intended generality--to rather rigid solutions, avoiding issues of learning and self-improvement. The problem of how to specify large amounts of complicatedly structured higher level knowledge by interaction with working NLU systems remained unsolved. Such inflexible schemes are not very convincing as models of human language processing, nor do they seem appropriate as a basis for developing practical systems, which should be able to adapt easily enough to the changing needs of really practical applications.

THE QUESTION-ANSWERING SYSTEM FAS2.5--GENERAL OVERVIEW

We give here a short overview of an experimental QAS FAS2.5, [3,5,6] developed recently for processing input sentences and answering questions in written German language. FAS2.5 is a multipurpose system, i.e. it was conceived as a well balanced general framework for restricted NL communication, adaptable to many different microworlds and fields of application.

To get a working system for special users or applications, additional field-specific knowledge is needed. The availability of such knowledge is very important. It must be filled into the general system, preferably in the form of NL statements. In this way, by augmenting an already existing, sufficiently general framework, it seems easier to create different application systems. But by no means do we claim that special problems may be solved solely by using general methods.

For a fixed purpose, the resulting system often does not need all the devices provided by the general system; therefore all irrelevant parts (procedures and data) should be omitted.

In the design of FAS2.5 we tried to avoid the danger of perfectionism. On the other hand we stressed issues of flexibility, generality, failure-tolerance, adaptability to specific user-requirements, and self-improvement by learning [4]. We adopted a sufficiently powerful scheme of knowledge representation similar to [2,11,13,14] and designed the input language so that the content of all that can be said in NL may be expressed in it--sometimes with minor deviations from the natural form of utterances.

Our system tries to understand (at least literally) almost all verbal sentences given to it. The input language covers a wide range from completely unchanged German sentences to more artificial looking, explicitly structured sentences of our version of "stylized" German (similar in spirit to [12]).

FAS2.5 has a very flexible strategy for analysis and interpretation of sentences. Successively more specialized and sophisticated semantic knowledge may be superimposed over some basic syntax-oriented default methods and always has priority, if it is available.

The system tries to learn as much as possible from the input texts given to it. Lexical, syntactic, semantic, factual, and general knowledge learned out of the given verbal input may be helpful for accomplishing related tasks in the future, e.g. understanding related sentences or answering questions.

The basic components and the information flow in the system are shown in Figure 1. The system can be roughly divided into three parts: input processing (language analysis), output processing (language generation), and internal processing (information retrieval including processes of inference).

The input language combines a nearly unchanged NL with the possibility to use stronger stylized language constructs whereever advantageous according to user needs and systems capabilities. (For instance, some artificial morphemes may be inserted into a German text for explicitly structuring sentences or marking syntactic categories of words for avoiding ambiguities.) The design philosophy and some details of the input language and the input processor of an earlier version of the system were described in [3] and [7]. In the last year the quality of the input processor was improved so much that now a great many unprepared German sentences may be accepted as input from the system [5].

Input texts in an approximated German language are sequences of declarative sentences, questions, and commands. The sentences of a text normally have a common universe of discourse and are interconnected by reference mechanisms. They are given to the system as strings of symbols and are transformed by an input scanner to simple lists containing as atomar elements the words and punctuation marks of each sentence in the original order. Linguistic processing of input texts is performed in several, partly overlapping steps, leading to different intermediate levels of representation. The linguistic processor includes ATN grammars (adopting the formalism of [20]), dictionaries, and procedures for morphological analysis (for German and English), a special scanner for NL text input, and different procedures for resolving references.

All factual and most of the conceptual (semantic) knowledge of the system is represented in a semantic network (SN). Network structures represent logically interpretable propositions and provide a very fast access between semantically related parts of the stored knowledge. New facts asserted as input sentences will be assimilated into the SN extending it in an incremental way. Conclusions drawn from asserted facts by spontaneous inferences also may be assimilated and explicitly stored in the SN.

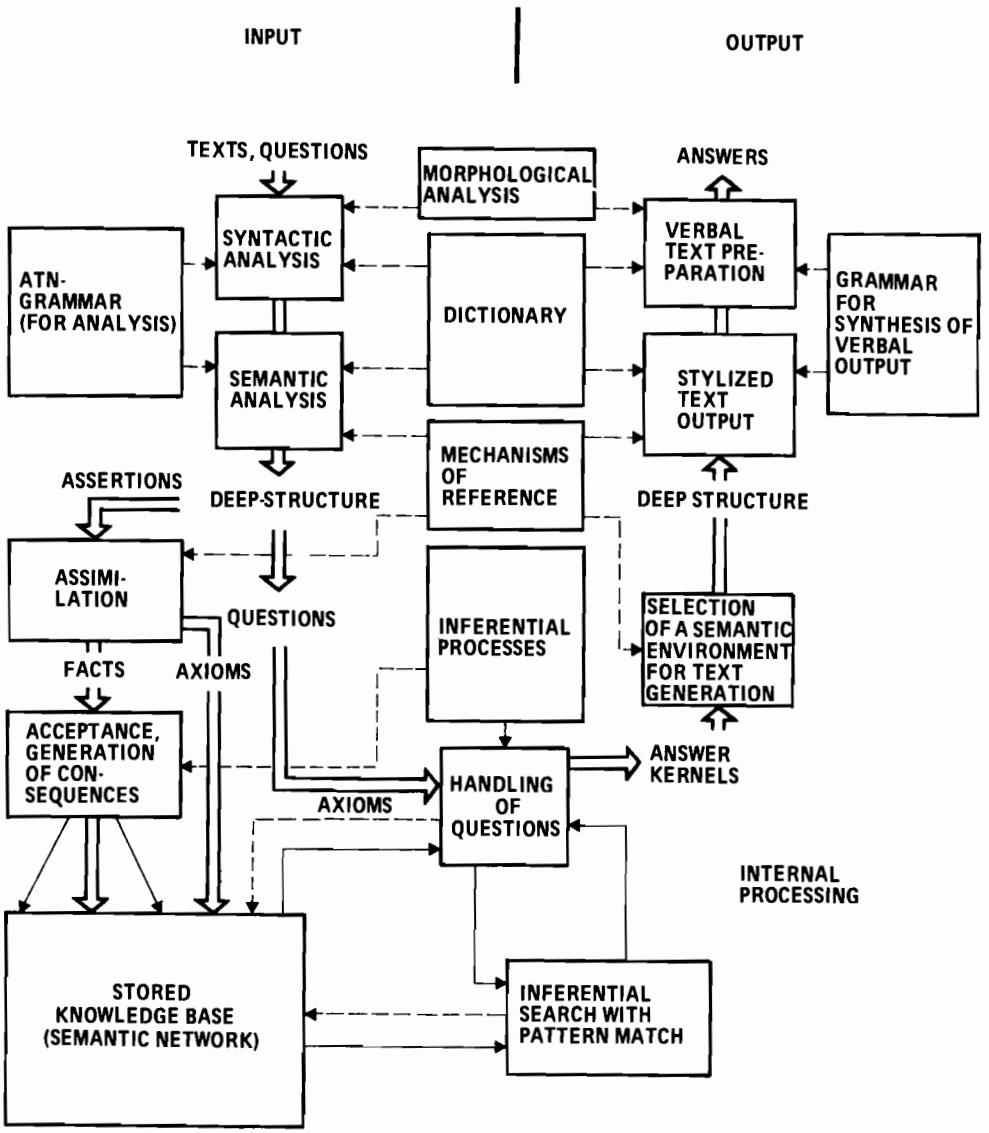


Figure 1. Knowledge sources and flow of information in the multipurpose German Language Question Answering System FAS2.5.

Questions are transformed to proposition schemes containing different kinds of variables. Answer finding as well as resolving of references is performed by restricted goal-directed (PLANNER-like) inference techniques supported by powerful pattern-matching procedures.

So far, the system has only limited capabilities for generation of stylized NL text output. It produces intelligible, but often not sufficiently (syntactically) polished verbal descriptions of answers found by the system.

Some classes of general propositions ("axioms" containing different quantified variables) may be recognized and correctly interpreted by the system. As logical implications or transformation rules they are represented in a special way and are very important for the inferential capability of the system.

NEWER IMPROVEMENTS IN FAS2.5

At the end of 1975 our QAS consisted essentially of the input processor described in [3] (which is also the core of the system FAS-2/FAS'75 [21]) and a component for deductive answer finding. The following improvements or extensions have been made since then:

- A large system of general concepts, arranged in a hierarchical way, was added to the system.

- Facilities for processing not only particular factual knowledge, but also generalized propositions expressed by verbal sentences were developed (definitions, meaning postulates, rule-like assertions describing temporal, causal, or motivational interdependencies between classes of situations or events).

- Techniques for goal-oriented inference (adopting the problem-reduction approach within a three-valued logic) have been improved and extended pattern-matching schemes, adapted to specific requirements of NLU and to the structure of our SN, provided.

- Refined methods for handling syntactic and semantic reference (pronouns, pro-adverbials, noun phrases attributed by relative clauses or other noun and prepositional phrases, subordinated clauses) have been added.

- Extensive morphological analysis for lexical units not contained in the stored dictionary now may be done by efficient procedures, tailor-made to the needs of a NLU-system.

- Mechanisms for generation of (more stylized) verbal output for describing internal objects that have no external name are now available.

- A complete linguistic processor (dictionary, ATN grammar, morphological analysis) exists now for restricted English as input language. It may be used instead of the originally developed component for handling German sentences.

LEXICAL ANALYSIS

Dictionary look-up, morphological analysis, and context-dependent guessing of lexical categories for unknown words of the input sentence jointly extract the basic forms of concept words. These are the smallest building blocks for constructing conceptual representations. We use a core dictionary of the most important basic words (containing about 1500 lexical entries) which can be arbitrarily extended, depending on the application field.

Because of the small size of our dictionary, procedures for morphological analysis are very important to find out the appropriate syntactic categories and basic forms of the words (in input texts) that are not included in the dictionary. The core dictionary at present contains mainly the structural or functional words (articles, pronouns, prepositions, conjunctions, some adverbials), but also the most important content words (including some important irregularly inflected word forms).

Uncommon words, if not included in the dictionary, may be marked by the user as belonging to a distinct word category. This is done immediately in the input text by putting a special symbol in front of the word in question. If a word is neither marked, nor identified within the dictionary, the system tries to generate a plausible hypothesis concerning the correct lexical handling of it by investigating characteristic features of the actual syntactic context and of the internal morphological structure (suffixes and inflectional endings) of the word.

Lexical hypotheses generated in this way can be inscribed into the dictionary and can prove helpful for analysis of other sentences later on. This type of lexical learning greatly improves the flexibility and self-adaptability of our system.

SYNTACTIC ANALYSIS

Syntactic analysis is controlled by a specially written ATN grammar for German sentences and performed with the aid of a general parser for ATN grammars. Traversing the sentence from left to right, all relevant information carried by a linguistic unit (e.g. a noun phrase or sentence) is gathered and stored in an association list (called "register-list") whose elements are attribute-value pairs of ATN registers.

Figure 2 shows as a directed graph the syntactic skeleton of a simplified version of our ATN grammar for German sentences; Table 1 explains the word classes appearing in Figure 2.

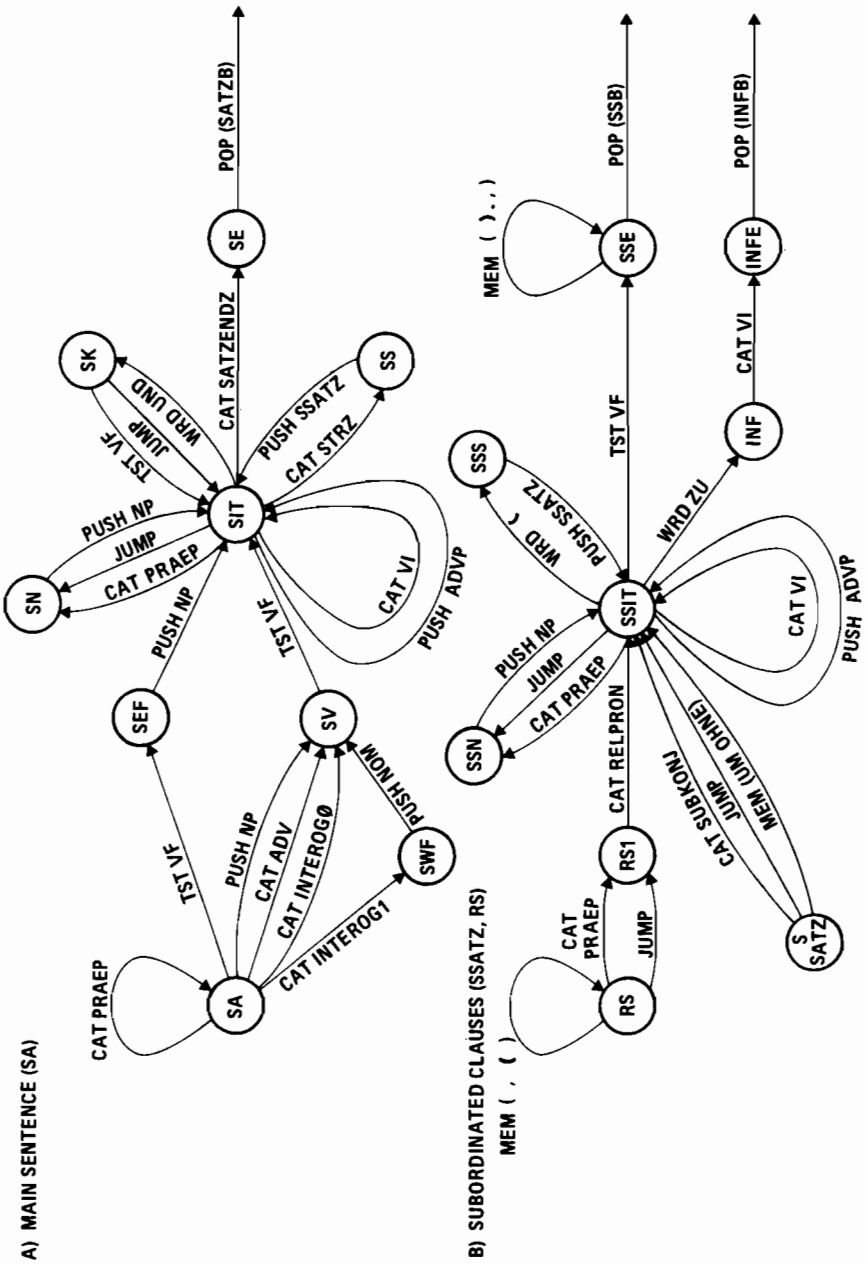


Figure 2. Simplified ATN-grammar for German sentences.

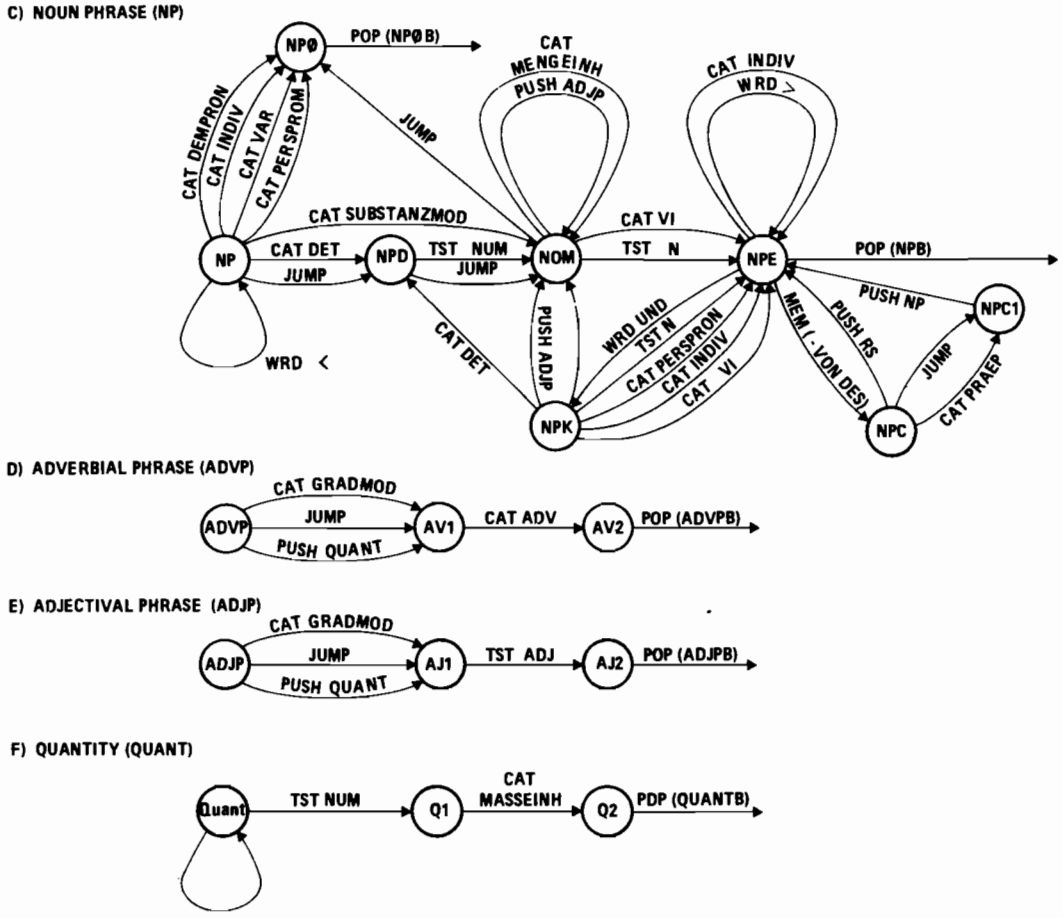


Figure 2. (continued)

By designing into the grammar look-ahead mechanisms for avoiding as far as possible superfluous syntactic ambiguities, simplifying the parser, and augmenting the input language by additional (facultative) means for clearer (more explicit) structuring of sentences, the whole process of syntactic analysis has been significantly streamlined.

Table 1. Word categories appearing in the ATN grammar (see Figure 2).

DET (determiner, article, quantifier), e.g. DER (the), JEDER (each)
PRAEP (preposition), e.g. IN (in), MIT (with), FUER (for)
SUBKONJ (subord. conjunction), e.g. WEIL (because), ALS (when)
NUM (number), e.g. 1, 23, 850, ZWEI (two), ZEHN (ten)
MASSEINH (unit of measurement), e.g. METER, JAHR (year)
MENGEINH (unit for physical quantities), e.g. TONNE (ton), FLASCHE (bottle)
GRADMOD (gradual modifier), e.g. SEHR (very), KAUM (scarcely)
QUANTMOD (modifier of abstract quantities), e.g. GENAU (exactly)
SUBSTANZMOD (modifier of abstract quantities), e.g. VIEL (much)
PERSPRON (personal pronoun), e.g. ER (he), SIE (she, they)
DEMPRON (demonstrative pronoun), e.g. DIES (this)
RELPRON (relative pronoun), e.g. DAS (that), DEM (whom)
INDIV (proper name), e.g. VICTOR, TURIN, LONDON
VAR (variable), e.g. X, Y, Z, MAN (everyman), NIEMAND (nobody)
N (noun), e.g. MENSCH (man), STADT (city), KRIEG (war)
VF (finite verb), e.g. IST (is), HATTE (had), GING (went)
VI (infinitive form of the verb), e.g. SEIN (be), GEHEN (go)
ADV (adverb), e.g. IMMER (always), SCHNELL (fast), DORT (there)
ADJ (adjective), e.g. GRUENE (green), ALTES (old)

SEMANTIC INTERPRETATION

From the register list produced by syntactic analysis of a linguistic unit, a treelike structure of embedded terms is built up after some additional semantic tests are passed. We call it "deep-structure" because it may be considered the result of applying normalizing linguistic transformations to plain syntactic structures. This structure may also be considered as context-independent linguistic meaning structure of an isolated sentence. Such a structure may be considered as a blueprint for the ultimate conceptual structure that must finally either be found in the actual knowledge base by resolving all references, or otherwise must be newly produced.

For instance, for different kinds of clauses and sentences (assertions, questions, and commands) unified (canonical) deep structures are produced by transformational procedures. The most important tenses, passive voice, articles, infinitive constructions, negation, and quantification are handled here in an appropriate way.

Our basic mechanisms for interpretation of sentences are rough syntax-oriented default methods. They allow correct interpretation of a lot of not too complicated German sentences. Successively more specialized and differentiating methods are superimposed, requiring more specific linguistic or factual knowledge. More specialized methods, if applicable, always have priority over less specialized ones.

ASSIMILATION AND REFERENCE

The further handling of deep structures in our system is different for simple facts (particular propositions), general sentences (general, nonlogical axioms), questions, and commands.

Deep structures describing objects, states, and events are mapped by assimilation and resolving of references (formally handled as term evaluation, but highly dependent on the actual context and the knowledge already stored) into our internal form of conceptual knowledge representation: logically interpretable semantic networks (SN). Only asserted propositions will be assimilated as a whole into the SN.

Assimilation may be accomplished in our system in two different moods:

- immediately after a deep structure for any syntactic unit was produced (before syntactic analysis of the whole sentence was completed), and

- after analysis of the whole sentence was completed.

Many words (e.g. pronouns) and descriptions (e.g. definite noun phrases) in NL are referring to items or concepts previously mentioned within the text or already stored in the memory. Mechanisms of reference in our input language--although more restricted and more clearly defined than in NL--provide not only useful means for more economical language use (by the user), but in many cases have to be considered as essential prerequisites for the verbal description of complicated ideas (that otherwise could not be communicated at all). Therefore, at least simple mechanisms for resolving references and handling context dependency will be required for advanced NLU systems.

We distinguish two kinds of reference: syntactic and semantic. Items introduced or mentioned within a few sentences uttered

immediately before the most important one may be referred to in an extremely short and simple manner--oriented strongly on syntactic criteria--by use of pronouns and pro-adverbs. Such criteria are the position of the referring word within the actual sentence, the position of a previously uttered language unit (taking into account the topic-comment distinction) that is considered as coreferent with it, and congruence of both constructs in gender and number.

More semantically oriented mechanisms of reference using definite noun phrases or some clauses allow reference to arbitrary entities mentioned at any time before or already stored otherwise in the knowledge base of the system. This kind of reference especially requires the activation of inferential processes. For simple cases only very basic relations of conceptual subsumption have to be considered. More complex noun phrases with other phrases or relational clauses as attributes have to be handled by means of the same inferential processes as are activated for answer finding in FAS2.5.

KNOWLEDGE REPRESENTATION

All factual and most conceptual knowledge of our system is represented in the unified form of a hierarchical SN whose nodes represent individual entities of different sorts (objects, locations, states, and situations), concepts (classes, properties, relations, or functions of objects, events, etc.), numbers, time moments, quantities, and different quantified variables.

The arcs of the network are labeled only by names of so-called basic semantic relations and can be followed immediately (by direct access) in both directions. They represent either defining or asserted basic relations. Defining outgoing arcs of a node normally explain what kind of thing or concept the given node represents. Asserted outgoing arcs of a node establish specific interrelationships between this node and others.

All concept nodes in the SN are arranged in a hierarchical way by connecting them by arcs labeled SUB (subordination or specialization of concepts). Figure 3 shows a fragment of the upper parts of this conceptual hierarchy that are important for arrangement of linguistic and general world knowledge and for many operations in the system (e.g. simple inference and resolving of references).

The basic semantic relations can be grouped together depending on the different sorts of their arguments. Predication (membership), certainly the most important relation holding between individual objects or events, and appropriate class concepts are represented by the relation IS. The negated form is ¬IS, the not-truth-asserting variants (subjunctive mood) are SEI and ¬SEI.

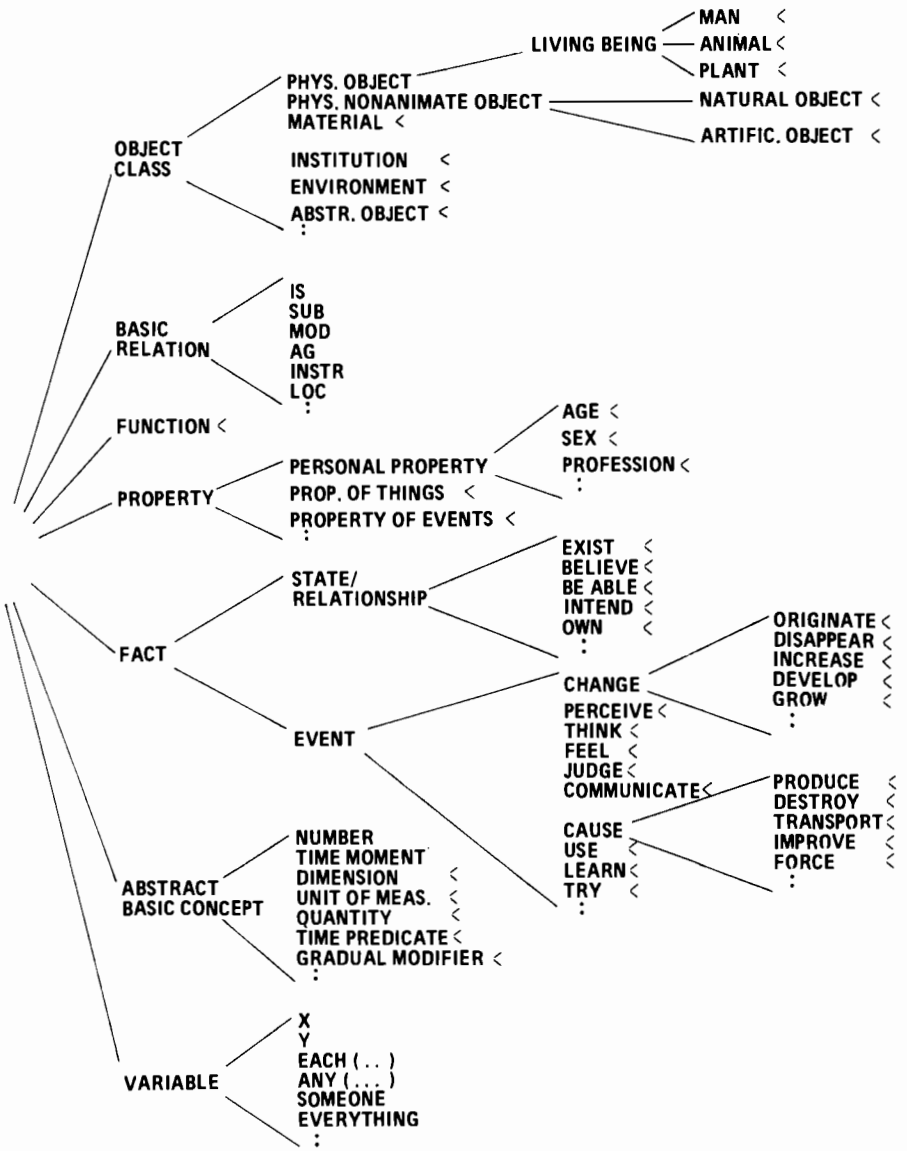


Figure 3. Conceptual hierarchy of stored knowledge (fragment).

Important relations between concepts are SUB (subordination or inclusion of concepts, subset relation), EXCL (exclusion), MOD (unspecific modification), GRAD (gradual modification), and QUANT (quantitative modification). Abstract quantities are represented by using MZ (number) and ME (unit of measurement). Relations for comparison of numbers, quantities, and time moments are GR (greater than), EQ (equal), and GREQ (greater than or equal). NUM relates to a set of objects ("compound object") the number of its parts. Deep case relations between an event and the entities participating in it are: AG (agent), RECIP (receiver), OBJ (object, experiencer), INSTR (instrument), DIR (direction, goal), ORIG (origin), ROL (role played by an entity), THEMA (object of communication or thought). The relation between an event and its place is LOC, between the time moment (or interval) MOM (MOMØ and MOM1 for the start and the end respectively).

Important basic relations between independently defined events or states are CAUS (causality relation), METH (method), INT (intention, purpose), and CIRC (circumstance). CONDIT (the if-then relation) is most important for representing general rule-like, inference-enabling propositions describing implicative relationships between classes of states or events. All other relations not included in the set of basic semantic relations may be represented as special concepts by nodes of the SN.

Simple quantified propositions (e.g. "Each dog is an animal."; "Everyone loves someone.") are representable within the SN formalism, as long as several universally quantified variables with different scopes do not appear and the scopes of all existentially quantified variables remain within the scopes of the universally quantified ones. More complicated general propositions (meaning postulates for words or concepts, and empirical rules) will be represented as transformational rules or implications (remotely resembling PLANNER theorems). They are represented as lists containing SN nodes and are also immediately accessible from appropriate concept nodes.

AN EXAMPLE

Consider now as an example the following short German input text that will be accepted by FAS2.5 and mapped into the semantic network.

> VICTOR IST EIN WISSENSCHAFTLER UND LEBT IN MOSKAU. VON DORT FLIEGT ER MIT EINEM SOWJETISCHEN FLUGZEUG DER FLUGGESELLSCHAFT AEROFLOT NACH WIEN, UM EINE KONFERENZ (DIE VON IIASA VERANSTALTET WIRD) ZU BESUCHEN. DAS FLUGZEUG IST EINE IL-62. ES HAT 158 SITZPLAETZE UND SEINE REISEGESCHWINDIGKEIT BETRAEGT 850 KM/H. WIEN IST HAUPTSTADT VON OESTERREICH. ...
>
> (Victor is a scientist and lives in Moscow. From there he is flying with a Soviet airplane of the airline AEROFLOT to Vienna to attend a conference organized by IIASA.

The airplane is an IL-62. It has 158 seats and its speed is 850 km/h. Vienna is the capital of Austria. ...)

This text describes a few simple facts ("The IL-62 has 158 seats."; "Vienna is the capital of Austria.") which could be contained in a relational data base. But--as most texts--it shows also much more complicated sentences with intentional constructions ("Victor intends to attend a conference."), time dependency, coordination and subordination of sentences, and anaphoric reference.

Figure 4 shows the representation of the factual knowledge contained in this text. It is produced by our system as a result of analysis and assimilation of the text. Of course, parts of the stored knowledge may also be given to the system in a more formal way, either as text in a more restricted and stylized natural language, e.g.

```
: VICTOR TUT MIT-INSTR E / SOWJETISCH * FLUGZEUG -
VON : AEROFLOT VON-ORIG : MOSKAU NACH-DIR : WIEN
V FLIEGEN .
```

or as a list of relational triplets or as a specially prepared positioned data file.

Now let us have a closer look at the different processing steps of the input sentence

```
VON DORT FLIEGT ER MIT EINEM SOWJETISCHEN FLUGZEUG
DER FLUGGESELLSCHAFT AEROFLOT NACH WIEN .
```

Lexical analysis for most words determines by dictionary look-up their syntactic categories and relevant syntactic features. By morphological analysis the infinitive FLIEGEN (fly) is found as the basic form of the finite verb FLIEGT--which is not in the dictionary--and likewise the adjective SOWJETISCH (Soviet) as the basic form of the inflected form SOWJETISCHEN. If the word FLUGGESELLSCHAFT (airline) is not yet known to the system, then by morphological analysis the suffix -SCHAFT (typical for German nouns) will be recognized and so this word will be classified as a noun.

Syntactic analysis then produces the following list of ATN-register values:

```
STYP (sentence type): AUSSAGESATZ (factual assertion)
WW (truth value): T (true)
NEG (negation of sentence): NIL (not available)
PASSIV (possibility of passive voice): NIL (not pos-
  sible)
```

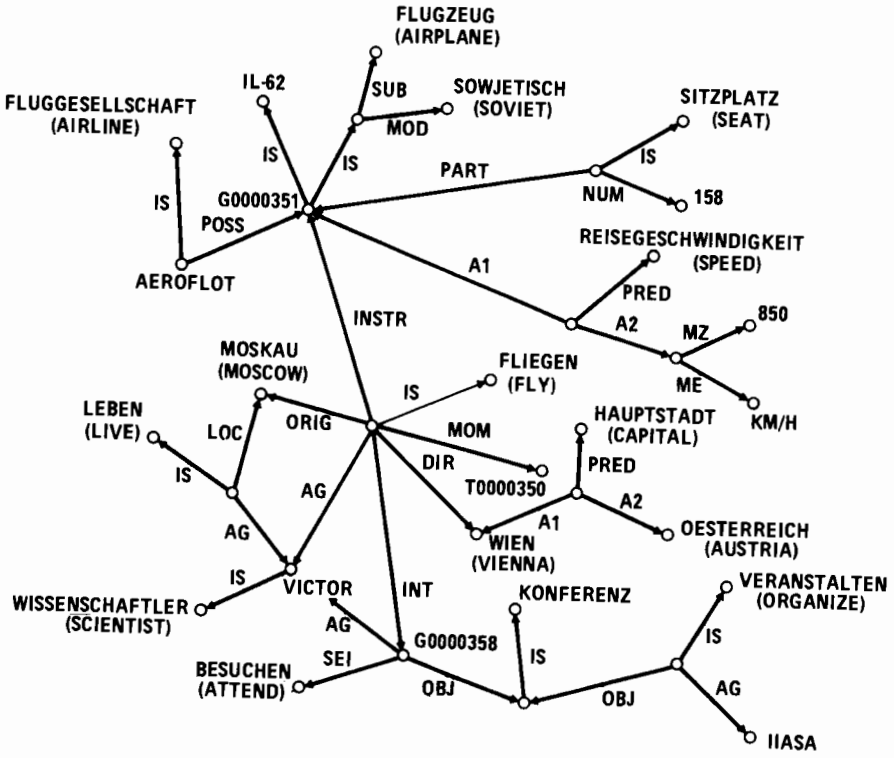


Figure 4. Fragment of factual knowledge in a semantic network.

```
GENERAL (sentence interpretable as general proposition):
  NIL
TEMP (tense):  PRES (present)
VF (finite verb dominating the sentence):  FLIEGEN (fly)
NPLIST (list of noun phrases):  ((ER))
VONLIST (list of VON-phrases):  ((DORT))
PPLIST (list of prepositional phrases):
    ((MIT (E (CONCEPT (SUB FLUGZEUG)(MOD SOWJETISCH))
             (ATTRIB (GENIT (ISTEIN AEROFLOT FLUGGES-
                                    SELLSCHAFT))) ))
     (NACH WIEN))
ADVLIST (list of adverbs):  NIL
```

Semantic interpretation and assimilation of substructures (including resolving of references: ER refers to VICTOR, DORT to MOSKAU) leads to the following deep-structure tree:

```
(AUSSAGESATZ
   (TEMP PRES)
   (AG VICTOR)
   (INSTR G0000351)
   (ORIG MOSKAU)
   (DIR WIEN) ) .
```

By assimilation of this tree a new event node G0000352 with its defining links to other nodes of the SN is created.

All substructures of our SN have a *logical interpretation* important for inference, answer finding, and text generation. Our sample sentence will be interpreted as a conjunction of some variable-free basic propositions

$$\begin{aligned} &IS(G0000352,\ fly) \wedge AG(G0000352,\ Victor) \wedge \\ &\quad INSTR(G0000352,\ G0000351) \wedge ORIG(G0000352,\ Moscow) \wedge \\ &\quad DIR(G0000352,\ Vienna) \wedge MOM(G0000352,\ T0000350)\ , \end{aligned}$$

where IS, AG, INSTR, ORIG, DIR, and MOM are two-place predicates, and their arguments are all constants here. Constants are concepts (e.g. fly) or individual entities that have (as Victor, Moscow, Vienna) or do not have external names (e.g. G0000351: Victor's airplane to Vienna).

For *internal representation* of SN the LISP language (with its unique atoms, each with its own property list) provides well suited facilities. We adopted the convention that all network nodes will be represented by LISP-atoms. Nodes representing unnamed concepts or entities will be given an artificial name (e.g. G0000352) by the GENSYM function of LISP. All (defining and asserted) outgoing (or incoming) arcs of a node are listed in its

property list by identifying their relational label and their goal (or source) node. Also the sort of each node (entity, variable, concept, proposition) is indicated in the property list.

QUESTION ANSWERING

Questions should be logically interpreted as proposition schemas containing variables. They correspond to more or less general descriptions of classes of propositions supposed to be contained in the knowledge base. Basic processes of question-answering are to find such asserted propositions in the stored data base that can be substituted under the questions' proposition schema (pattern) and to retrieve (parts of) them in different ways. For attaining a high efficiency of question-answering it is important to have sophisticated mechanisms for pattern matching.

In FAS2.5 the information search with pattern matching originates from the question pattern and proceeds in a selective way. Pattern-matching is done rather efficiently paying attention to the special requirements of NL descriptions and to the chosen knowledge representation with a hierarchical arrangement of concepts and different kinds of quantified variables within the SN. The power of our pattern-matcher is significantly increased by incorporation of specially designed procedures for performing the most important (apparently ubiquitous) deductive operations (as following hierarchical chains between concept nodes in both directions, handling symmetry and transitivity of basic relations, and looking for different possible representations of relevant facts).

Closed questions simply ask whether any propositions are known that are instances or counterexamples of the implied proposition scheme. In principle, they have to be answered by one of the three "formal" answers: "yes", "no", and "unknown" (which surely does not meet the user requirements for a more comfortable dialogue mode).

Open questions ("wh-questions") introduced by interrogatives imply proposition schemes containing (besides other variables) the so-called question-variable (symbolized "?") and are much more diversified. They are asking not only for the existence of proposition instances matching the pattern of the question, but--if such instances are found--cause the retrieval of some or all entities that are involved in or related to these proposition instances in a distinct way. The formal answer to an open question is a (possibly empty) set of such items that matched the question variable within successful matches between stored propositions and the question pattern. The position of the question variable in the question pattern indicates what special kinds of information are wanted by the question.

The following questions can be posed to FAS2.5 and answered very fast, simply by extended pattern matching after the input text on Victor's flight to Vienna (see above) was "understood" by the system.

Q. WO LEBT VICTOR ?
(Where is Victor living ?)

A. IN MOSKAU
(in Moscow)

Q. WOHIN FLIEGT VICTOR ?
(Where is Victor flying ?)

A. NACH WIEN
(to Vienna)

Q. WOZU KOMMT ER DORTHIN ?
(For what does he come there ?)

A. UM E KONFERENZ ZU BESUCHEN
(to attend a conference)

Q. BESUCHT VICTOR DIE KONFERENZ - VON IIASA ?
(Does he attend the conference of IIASA ?)

A. UNBEKANNT
(unknown)

Q. WER FLIEGT ZUM BESUCHEN - EINER KONFERENZ NACH WIEN ?
(Who is flying to Vienna for a visit of a conference ?)

A. VICTOR

Q. MIT WELCHEM FLUGZEUG FLIEGT ER ?
(With what airplane does he fly ?)

A. MIT/ E IL-62
(with a IL-62)

Q. WELCHE REISEGESCHWINDIGKEIT HAT SIE ?
(What speed does it have ?)

A. 850 KM/H

Q. BESUCHT VICTOR DIE KONFERENZ ?
(Is Victor attending the conference ?)

A. UNBEKANNT
(unknown)

Q. WIEVIELE SITZPLAETZE HAT DAS FLUGZEUG , MIT DEM VICTOR NACH DER HAUPTSTADT VON OESTERREICH FLIEGT ? (How many seats has the airplane that Victor is flying to the capital of Austria in ?)

A. 158

For the sake of clarity we restricted the factual knowledge in the SN available for this demonstration to only a few propositions. So it was not possible to demonstrate answering of questions that require a list of all stored entities of a certain kind to be found (e.g. "Which cities outside of USSR are connected by direct flights with Moscow?"). But notice the difficulties with such questions that may result from the fact that a sufficiently complicated knowledge base containing very heterogenous knowledge always has to be considered to be incomplete!

Although the questions shown here sometimes differed remarkably from the input sentences containing the knowledge necessary for answer-finding, so far we have only demonstrated the capabilities of FAS2.5 that are based on pattern matching together with linguistic normalization and resolving of references.

A much broader range of questions will become manageable by activation of goal directed deductive mechanisms. They are able to interact with arbitrary field-specific general knowledge that may be encoded in FAS2.5 in the form of quantifier-free implications or pattern-transformation rules that can be produced by FAS2.5 from German sentences [5] that are interpreted as "general" (e.g. "all" sentences, "if-then" sentences).

For questions that proved not to be answerable immediately by pattern-matching, these deductive mechanisms cause recursively the transformation into subquestions that it is hoped can be more easily manageable. Deductive mechanisms in the kind of problem reduction or searching AND-OR trees already had proved appropriate in the author's 1971/72 QAS FAS-1 [8] for processing simple queries for retrieval of a relational data base and will not be described here in detail.

For extending the inferential power of a practical system and its adaptability to different application areas or users, methods for recognition, interpretation, representation, and activation of generalized knowledge given to the system as NL sentences are very important.

REFERENCES

[1] Bobrow, D.G., and D.A. Norman, Some Principles of Memory Schemata, in D.G. Bobrow and A.M. Collins, eds., *Representation and Understanding*, Academic Press, New York, 1975.

[2] Hendrix, G.G., *Expanding the Utility of Semantic Networks Through Partitioning*, presented at the 4th IJCAI, Tbilisi, August 1975.

[3] Lehmann, E., Input Processing in a German Language Question-Answering System, in *Conference on Artificial Intelligence: Question-Answering Systems*, CP-76-6, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

[4] Lehmann, E., Self-Improvement of Natural Language Understanding Systems by Learning, Paper submitted for IJCAI-77, Cambridge, Mass., 1977.

[5] Lehmann, E., Ein Frage-Antwort-System mit variablem Diskursbereich und genereller Wissensrepräsentation, in L. Bolc, ed., *Natural Language Communication with Computers*, Warszawa, 1977.

[6] Lehmann, E., *Computersimulation des Verstehens natürlicher Sprache*, presented at Leopoldina-Symposium Naturwissenschaftliche Linguistik, Halle/Saale, July 1976. (also *Nova Acta Leopoldina* 1977).

[7] Lehmann, E., and F. Zänker, *Fakteneingabe für ein Frage-Antwort-System*, Bericht Nr. 156/111/9/74, VEB Robotron, ZFT, FG Grundlagenforschung, Dresden, 1974.

[8] Lehmann, E., *Ein Frage-Antwort-System mit Deduktions- und Lernfähigkeit*, VEB Robotron, ZFT, Dresden 1972; also in F. Klix, W. Krause, and H. Sydow, eds., *Analyse und Synthese von Problemlösungsprozessen II*, Kybernetik-Forschung, Vol. 5, Dtsch.Verl.d.Wissenschaften, Berlin, 1975.

[9] Lesser, V.R., R.D. Fennel, L.O. Erman, and D.R. Reddy, Organization of the HEARSAY-II Speech Understanding System, *IEEE Trans. Acoust. Speech, Signal Processing*, 23 (1975) 11-23.

[10] Montague, R., The Proper Treatment of Quantification in Ordinary English, in J. Hintikka, J. Moravcsik, and P. Suppes, eds., *Approaches to Natural Language*, Reidel, Dordrecht, 1973.

[11] Norman, D.A., and D.E. Rumelhart, *Explorations in Cognition*, Freeman, San Francisco, 1975.

[12] Palme, J., *A Simplified English for Question Answering*, Report C8256-11(64), Institute of National Defense, Stockholm, 1970.

[13] Palme, J., *The SQAP Data Base for Natural Language Information*, Report C8376-M3(E5), Försvarets Forskningsanstalt, Stockholm, 1973.

[14] Sandewall, E., Formal Methods in the Design of Question-Answering Systems, *Artificial Intelligence*, 2 (1971), 129-145.

[15] Sandewall, E., *PCF-2, A First-Order Calculus for Expressing Conceptual Information*, Uppsala University, Computer Science Dept., Datalogilaboratoriet, Uppsala 1972.

[16] Schank, R., ed., *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.

[17] Schubert, L.K., Extending the Expressive Power of Semantic Networks, *Artificial Intelligence*, 7 (1976).

[18] Walker, D.E. et al., *Speech Understanding Research*, Annual Report, Project 3804, Artif. Intell. Center, Stanford Research Institute, Menlo Park, California, 1975.

[19] Winograd, T., *Understanding Natural Language*, Edinburgh University Press, Edinburgh, 1972.

[20] Woods, W.A., Transition Network Grammars for Natural Language Analysis, *Comm. ACM*, 13, 10 (1970), 591-606.

[21] Zanker, F., H. Böttger, H. Helbig, E. Lehmann, and P. String, *Informationsdarstellung in einem Frage-Antwort-System*, presented at Internat. Symposium MKÖ, Techn. Univ. Dresden, Oct. 1975.

Use of Semantic Networks for Information Retrieval

G. Rahmstorf

INTRODUCTION

The information contained in so-called formatted data base systems is stored in fields of explicitly defined length. Several data structures such as hierarchical, network, or relational data structure are used to represent associated information [1]. Commonly defined data names are used to get access to stored information.

This paper is concerned with free text retrieval systems that store unchanged text of any kind--for example, abstracts of documents--as information for the user. Text retrieval is the more general term, whereas document retrieval indicates the main application field.

A survey on document retrieval was given recently by Salton [2]. Martin and Parker [3] have compared several features of text retrieval systems currently available.

Data base systems and text retrieval systems have been developed independently. The development of natural language (NL) interfaces for data base systems might help to bridge the gap between the two types of systems. Data structures used in formatted data base systems, text retrieval systems and NL question-answering systems (QAS) are compared in Figure 1.

The original information to be stored in both types of data base systems is communicated verbally or written as a text (A). This text can either be stored directly in a free text retrieval system and/or translated into a data structure (B) of a formatted data base system. Many attempts have been made to translate free text into a semantic network (SN) automatically (C). An SN is another type of formatted data structure that can be transformed into conventional formatted data files and vice versa.

To avoid sequential search through the whole text data base, most text retrieval systems prepare an additional inverted file, which is a list of all word occurrences (except user-defined stop words such as "the", "is", "for"). The list of words (D) is a Boolean expression in productive normal form (word A AND word B AND word C...).

The advantage of this technique is that the system can represent any text simply by the words of the text. But, on the other hand the Boolean expression is not a semantically precise document description.

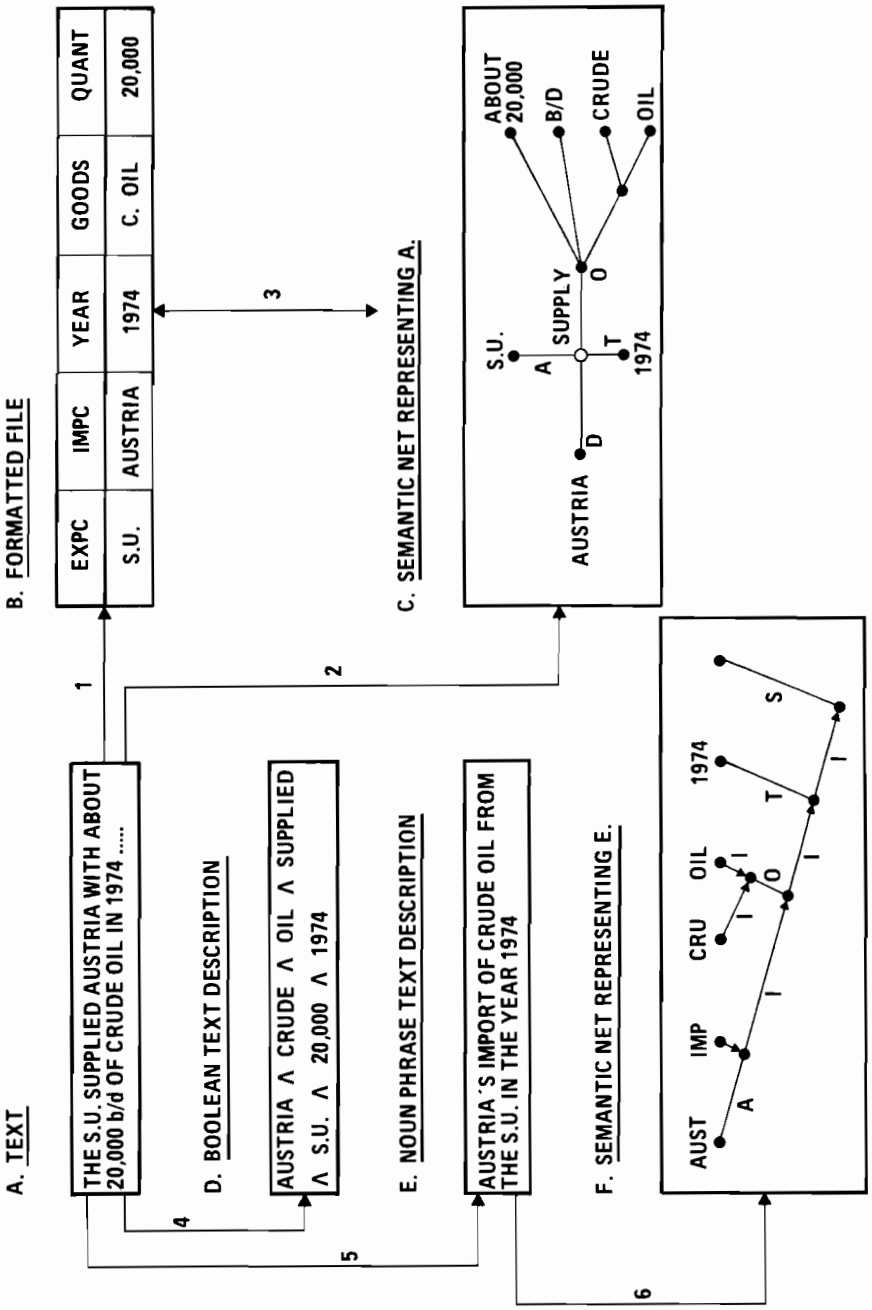


Figure 1. Representation of text by several structures.

In this paper we propose to describe a document by a special NL form instead of a Boolean function. This natural form is called noun phrase (E) and is represented as a node in an SN (F).

PROBLEMS OF CURRENT TEXT RETRIEVAL SYSTEMS

We base this proposal on a fundamental criticism of current text information retrieval systems. Several functions of such systems can be improved, but there are two main problem areas that require a substantial new approach for improvement.

Insufficient Precision and Recall

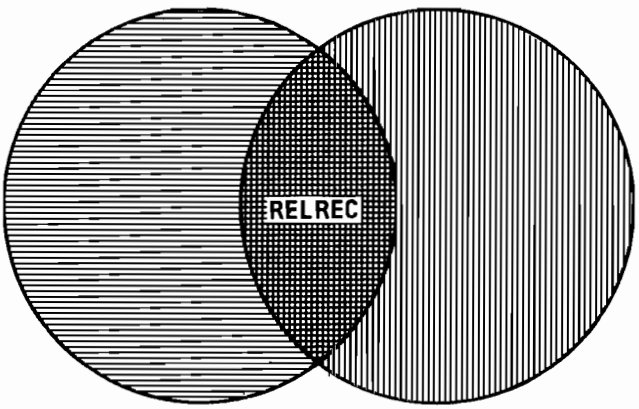
Responses given to a user's query are imprecise, because not all recalled documents are relevant. Responses are also not complete because not all relevant documents are recalled. The parameters commonly used to define precision and recall [4] are illustrated in Figure 2. Precision as well as recall are dependent on the type of stored information and the user's evaluation of the system response, but a general experience is that about 50% precision and recall are typical values for such systems.

Need for Expert's Assistance

The second deficiency is that the use of these systems requires a large amount of skill to describe the user's information request in the language of the system. "Most searchers today are information specialists who through constant practice are comfortable with the terminal, command language and data bases" [2] (Figure 3).

Many documentarists and librarians are not satisfied with automatic document description based on word occurrence in the text. They prefer to select intellectually additional descriptors from a thesaurus which are then associated with the stored document to improve the results of retrieval. This additional indexing work is a second source for the growing expense of information retrieval.

Several studies have been made to improve automatic indexing [5,14]. The objective of automatic indexing is primarily to "assign appropriate identifiers capable of representing information content to stored documents and incoming user queries" [6]. However, the set of assigned identifiers is not structured by a natural language syntax forming a phrase. As far as phrases are used to describe a document, they are taken from the text or from a list of phrases in a thesaurus. It seems to be extremely difficult to generate automatically a phrase representing the contents of the text. The extraction of phrases from an abstract was recently discussed in [7].



RELEVANT INFORMATION — RECALLED INFORMATION

$$\text{PRECISION} = \frac{\text{RELREC}}{\text{REC}} \times 100 \qquad \text{REL} \;/\; \text{REC} \;<\; 100\ \%$$

$$\text{RECALL} = \frac{\text{RELREC}}{\text{REL}} \times 100 \qquad \text{REL} \;/\; \text{REC} \;<\; 100\ \%$$

Figure 2. Definitions: precision and recall.

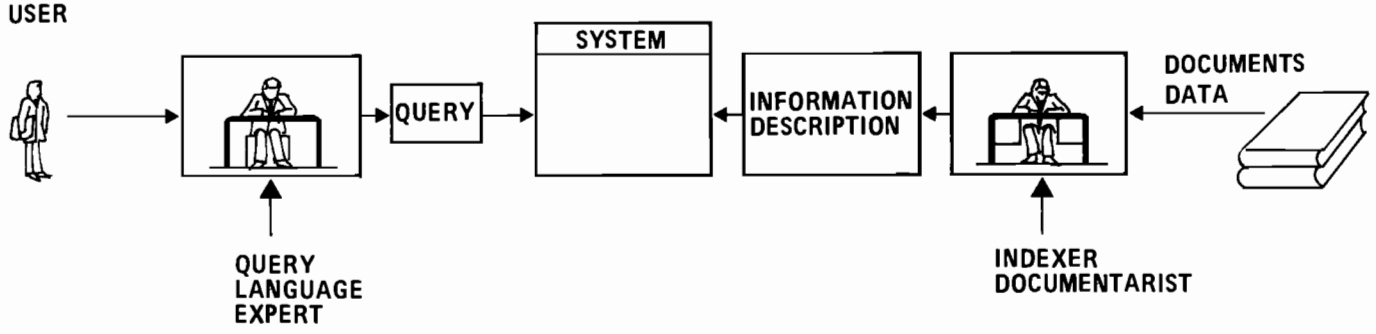


Figure 3. Efficiency problem: query formulation and information description.

Storage Requirements for Document Description

Text retrieval systems need much storage to represent the original text and the additional descriptive information automatically derived from the original text. In Figure 4 a simplified representation of the data structure of an advanced text retrieval system is shown [8]. The original documents (text) are stored in a text file that is accessed via a text file index. To avoid sequential search through all documents for each query, the system uses two additional files: a dictionary and an inverted file. The dictionary contains all words of the text, with pointers referring to records of the inverted file. The inverted file is mainly a list of occurrence entries for words. A word occurrence is described by its location within the document. The purpose of this context specification is to allow queries with contextual operators such as:

A ADJ B	words A and B are adjacent to each other in this order in the same sentence;
A WITH B	A and B occur in the same sentence; and
A SAME B	A and B occur in the same paragraph.

On the other hand, the additional descriptive information about the position of a word within a sentence, paragraph and document increases the storage requirements. This is again a reason to propose a shorter and more informative document description.

Performance Considerations

One major advantage of free text retrieval systems is that they generate the descriptive inverted file automatically. However, this generation process is done only once for each document. The expenses for document description charged to one user's search can therefore be divided by the number of searches. But, each user has to cover completely the total expenses for his own search process. Therefore, one must try to reduce the computer time needed for the search. One way is again to develop a more compressed document description.

Origin of the Problem

There are two main reasons for insufficient precision and recall.

Imprecise Language for Query and Document Description

Text retrieval systems use Boolean operators to describe the stored documents and to formulate the user's information requests.

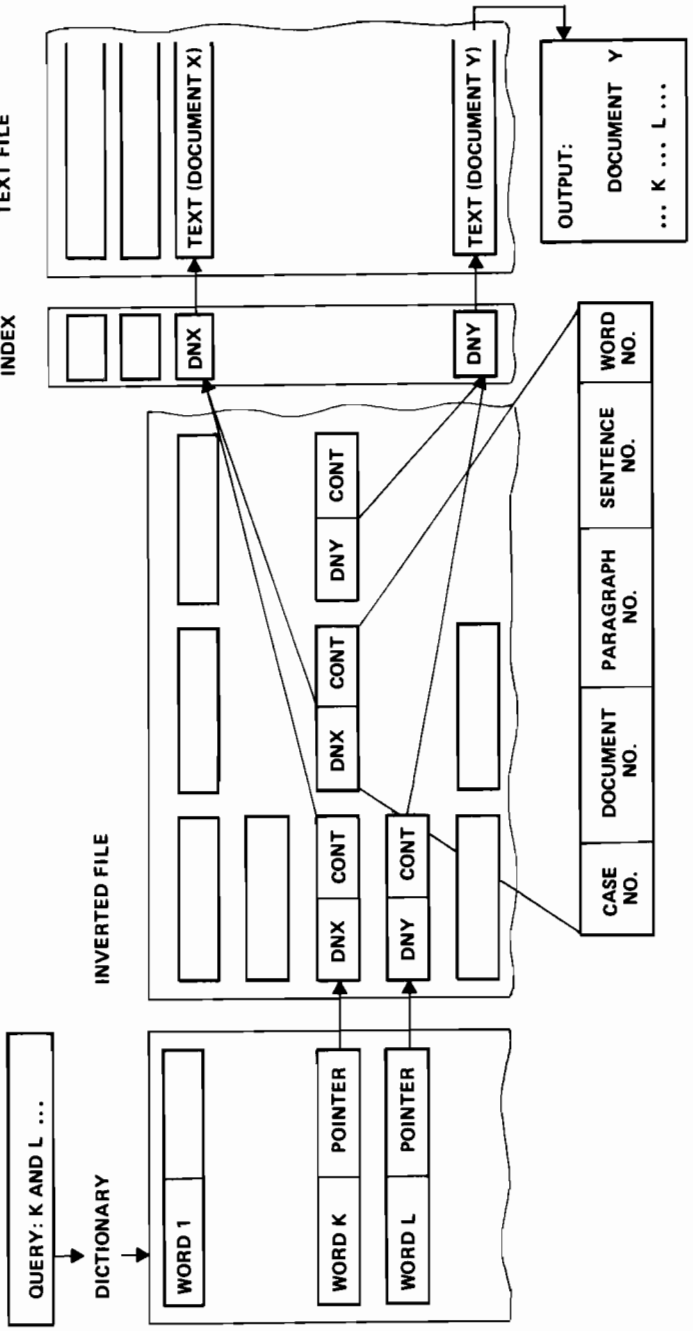


Figure 4. Storage structure of a text retrieval system.

Some query languages include other features such as the context specifications we have already mentioned.

Boolean expressions interpreted as a description of the document or as the subject someone likes to know more about, are ambiguous. A string of keywords connected by AND can represent many different concepts. What is the content of documents described by the following Boolean expression:

> "BASIC AND COMPONENT AND CONSUMPTION AND CONVERSION AND ELECTRICITY AND ENERGY AND MACHINES AND PRIMARY AND PRODUCTION"?

Because of the complete lack of syntax, we cannot decide on any specific interpretation of such an expression.

Wrong Usage of Query and Document Description Language

The reason for incomplete and unprecise system output is also a result of wrong usage of the available language. Specifications of Boolean expressions and context operator requires some training. Any user who is not well trained can easily make errors in query formulation or document description. An expert of the system must therefore translate the user's request into a query for the system, which is a new "major source of failure", because now one human being has to communicate his information need completely and accurately to a second person. A large percentage of the searching failures in 300 searches was attributed to inadequate user-system interaction [13].

PROPOSAL

To avoid misinterpretation of Boolean queries and document descriptions, several proposals have introduced additional features for the documentation language, such as role indicators or descriptor links [9]. They help to reduce the scope of interpretations for a Boolean expression, but these language features require more expert assistance and more indexer's time to describe the documents. The higher complexity of the artificial syntax for documentation languages decrease the cost-performance ratio of the whole system.

We need a language that is both less ambiguous and easier to learn. Therefore we propose to use a few basic syntactical structures of the NL for query and document descriptions. We recommend restricting the NL to basic noun phrases. A noun phrase can be defined by a formal grammar of context free production rules using the word classes noun, adjective, and preposition. The following English expressions are noun phrases:

- Information system design process;
- Flexible, dielectric materials;
- Effects of pressure on the optical properties of a polymorph of germanium.

Noun phrases represent the natural form in which the human communication partners express an information request. Noun phrases are also the syntactical form of titles by which technical or scientific documents are usually described.

A title in noun phrase form must be precise and complete. If a given title of a document does not satisfy these requirements, it must be completed by another title or additional titles created by the author or indexer.

The former example for a Boolean document description is originally a title or a request of the following noun phrase form:

"Energy consumption for the production of basic components used for conversion of primary energy into electricity".

The noun phrase in this example is only one possible interpretation of the Boolean expression, which is ambiguous if interpreted as information requests or concepts. The transformation from Boolean to noun phrase form requires additional information represented by word sequence, special particles as prepositions, and inflectional forms. This additional structure information has to be analyzed to understand the noun phrase. Even if the result of such an analysis is limited by our current linguistic methods, it is better than a Boolean coordination of descriptors.

The system we propose should have the following functions for document description and document retrieval:

- An indexer or data base administrator may describe documents or data by noun phrases. These noun phrases are analyzed by the system and represented in a semantic network as a concept node. The system generates a unique number for the document description, which is then added to the document number.
- A user may ask for any information by formulating his query as a noun phrase. This noun phrase is analyzed by the system and represented as a node of the semantic network. The system analyzes which document or data descriptions are relevant to the query and generates a secondary Boolean query, including the relevant document numbers which are then used to retrieve the actual document.

STRUCTURE OF A NOUN PHRASE SYSTEM

These functions can be implemented by a system consisting of the following components:

- Noun phrase analyzer;
- Lexicon;
- Semantic network;
- Relevance analyzer;
- Document description file; and
- Text data base.

The text data base system for the storage of the documents can be a text retrieval system as currently available. The noun phrase system described here can be viewed as a front-end system used as an additional module for the text retrieval system to process document descriptions and queries in a higher language.

The Noun Phrase Analyzer (Figure 5)

The noun phrase analyzer is a linguistic processor that represents the meaning of given input phrase as a node of the SN. It consists of two interrelated submodules: a syntax parser and a semantic interpreter.

A set of context free grammar rules is used to analyze a given phrase. As a result of this analysis one or more syntactical tree structures are generated.

The semantic interpreter uses a set of rules to map nodes of the syntax tree into nodes of the semantic network. The linguistic methods and problems involved in noun phrase analysis are not discussed here.

Dictionary

The noun phrase analyzer needs a dictionary containing all nouns and adjectives used in noun phrases. The dictionary specifies for each word syntactic attributes and a pointer to the SN defining the meaning of a word as a concept within a conceptional network.

The Semantic Network

The SN is the central component of the information retrieval system. It is used to represent concepts and relations between concepts.

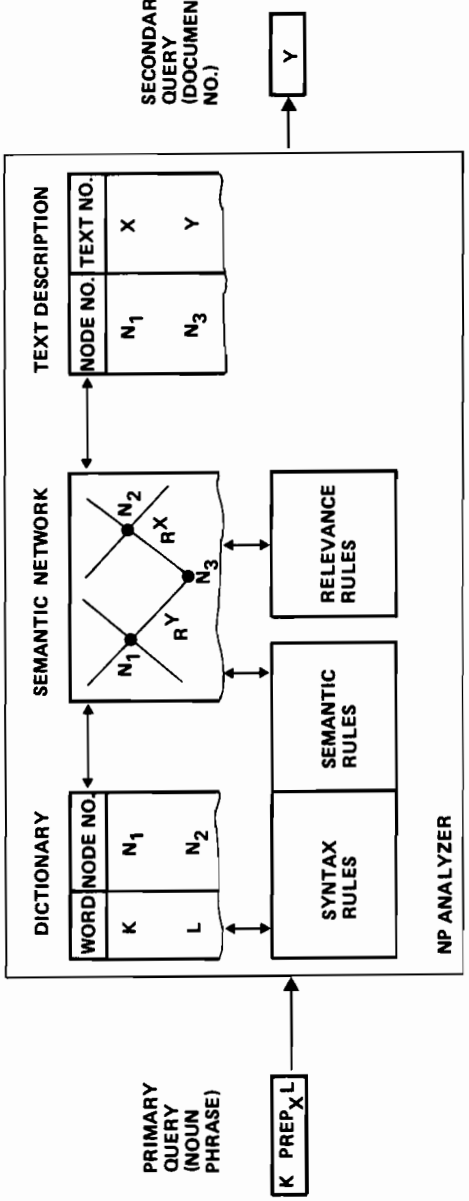


Figure 5. Components of a noun phrase retrieval system.

A concept can be expressed by a noun phrase of a language in English or German. Because noun phrases include also single nouns the concept network is also used to define the meaning of the nouns (and adjectives) of the dictionary.

Each concept of the semantic network can be used to represent a text (title of a document) or a search request. Not every possible noun phrase or document description is represented in the concept network. The number of possible concepts is immense, possibly infinite, because one can generate phrases recursively. But only those nodes in the concepts network that represent a document description for an actual document have to be generated. The nodes describing a document have to remain in the network only for the time the document is stored in the data base.

Concepts are nodes of the SN, relations between concepts are arcs between nodes. Information from the SN is used by the grammar rules to analyze noun phrases. The meaning of the analyzed noun phrase is represented as a node of the SN. The node representing a noun phrase can already exist or can be created by using relations that connect the new node to the existing nodes. The SN is used by the relevance analyzer to determine relevance between nodes representing text description and nodes representing a user query.

The SN can be viewed as an extended on-line thesaurus for an information system with additional functions. It is an extended thesaurus, because it contains not only technical terms, but also common language words. Classical thesaurus uses only a few relations [10] between terms such as:

- Narrower term,
- Broader term,
- Related term.

In the SN one needs more relations and more specific relations, similar to the case relations introduced by Fillmoore [11] and used by many NL systems.

The SN is a dynamic structure as only the concepts represented by dictionary words are stored permanently. Concepts representing complex queries are only created for the time when the requested phrase must be answered. Concepts representing document descriptions are only kept during the time the document is available in the data base system.

The SN is functionally more than only on-line thesaurus, because it is primarily not used to assist the user in generating a proper query, but to determine numbers of documents relevant to the primary query. These document numbers are taken from the document description file and are included in the secondary query.

Text Description File

The information unit stored in the data base system and described by use of the SN is a text. Each text is identified by a text number (or document number). Each document must be described by a title in noun phrase form. Each title or text description is represented as a concept node of the SN. Those concepts representing text descriptions are called T concepts. The text description file (Figure 4) contains all document numbers and numbers of T concept nodes representing the document description. The text description file is created and updated by the noun phrase analyzer by using the document number and text description as input.

Relevance Analyzer

T concepts representing titles of documents stored in the data base and Q concepts representing user requests are mapped as nodes of the SN. The relevance analyzer selects those T concepts relevant to a given Q concept.

As already discussed in the context of precision and recall, relevance is a very fundamental idea used to evaluate output of retrieval systems. But, so far, any evaluation is based on subjective decisions of the users of whether a given document is relevant for them or not. There is no way to check such a decision objectively.

If query and document description are both represented as nodes of an SN, one can define objective criteria for relevance. In Table 1 we have given some examples of such relevance criteria.

Relevance criterion 1 would be the simplest case, specifying that only those noun phrases represented by the same concept are relevant for a given query. This means that each synonymous syntactical construction used as a title of a document is classified as a relevant title.

Another relevant criterion generally accepted, specifies that more important concepts are classified as relevant. If a user asks for "energy consumption in Austria", the system would classify "Austrian energy consumption in 1970" as a relevant document, because the title has an additional time attribute which makes it "more special" (subconcept) according to the relevance criterion 2. Relevance between concepts represented in an SN is a new field that is not discussed completely at this point. However, the proposed approach allows definition of formal relevance criteria. The user can now have better control of system recall by specifying his own "hard" or "soft" relevance criteria depending on the kind of information need.

There are two typical information need patterns (Table 2). Noun phrases used as query and document description language seem to be more appropriate for specific information requests rather than for a general overview.

Table 1. Relevance specifications.

A User's Information Request Q Concept	Requested Relevance Criteria	Relevant Documentation/Data Noun Phrases T Concepts
Energy Consumption in Austria	1. Same concept	Austrian consumption of energy. Consumption of energy in Austria.
Energy consumption in Austria	2. More special concepts	Oil consumption in Austria Austrian energy consumption in 1970
Energy consumption in Austria	3. Related concepts	A model of the Austrian energy consumption. Impact of the oil crisis on the energy consumption in Austria.
Energy consumption in Austria	4. Associated concepts	Austrian energy production. Conversion of primary energy in Austria.
Energy consumption in Austria	5. More general concepts	Energy consumption. Consumption in Austria.

Table 2. Information needs.

Overview, Awareness of Working Environment	Special Problem Solution Interest
1. Type of User Query	
What is the current structure of my scientific field?	What is the answer to my specific question?
Most important problems, theories, and results?	All relevant facts, data or text
Spread of papers on topics, terminology, actual trends	
2. User Behavior	
User is developing his own opinion by browsing through reference	User is carefully studying all relevant data and documents
3. System Properties	
Reply with a representative set of actual reference is sufficient	High recall is essential
	High precision is required
High precision and recall is not essential	
4. Information Description and Query Method	
Boolean description is sufficient	Noun phrases (or equivalent artificial description) or user known data/document names are necessary

Secondary Query

When the relevance analyzer has selected those T concepts relevant to the query concept, it takes the associated document numbers from the text description file and creates the secondary query. The secondary query is a list of the relevant document numbers connected by Boolean operators OR. The secondary query is given as input to a standard text information system to retrieve the text from the data base.

Comparison

After having discussed the major components of an information retrieval system based on a semantic network, Figure 6 describes how the components interact for both text description and search. Comparing this structure with standard text retrieval systems that use Boolean methods, we see that the problem fields resulting in insufficient recall and precision are replaced. For the document description function (right side of Figure 6) we expect the following improvements:

- Less ambiguous text description language (2);
- Easier text description (indexing) by the documentarist (6);
- More precise semantic representation (3).

Query analysis is expected to be improved because of:

- Less ambiguous query language (1);
- Simple, natural query formulation by the user (5);
- More precise semantic representation (3);
- Precise relevance criteria (4).

Precision is expected to be greater than with Boolean techniques. Recall can be controlled by the user with relevance specifications. Expenses for user assistance will be reduced, because noun phrases are a part of the user's NL.

Other Applications

The proposed technique for information description is not restricted to textual information. Noun phrases are the general tool of NL to identify any objects, entities or concepts. Therefore, one can also use noun phrases to refer to formatted data, too. Figure 7 indicates how subsets of a sample file of three records can be described or requested by use of noun phrases. For example, the noun phrase "Sales by Jones" is a description of only those records, where "Jones" is the data item in a data field characterized as "agent" field.

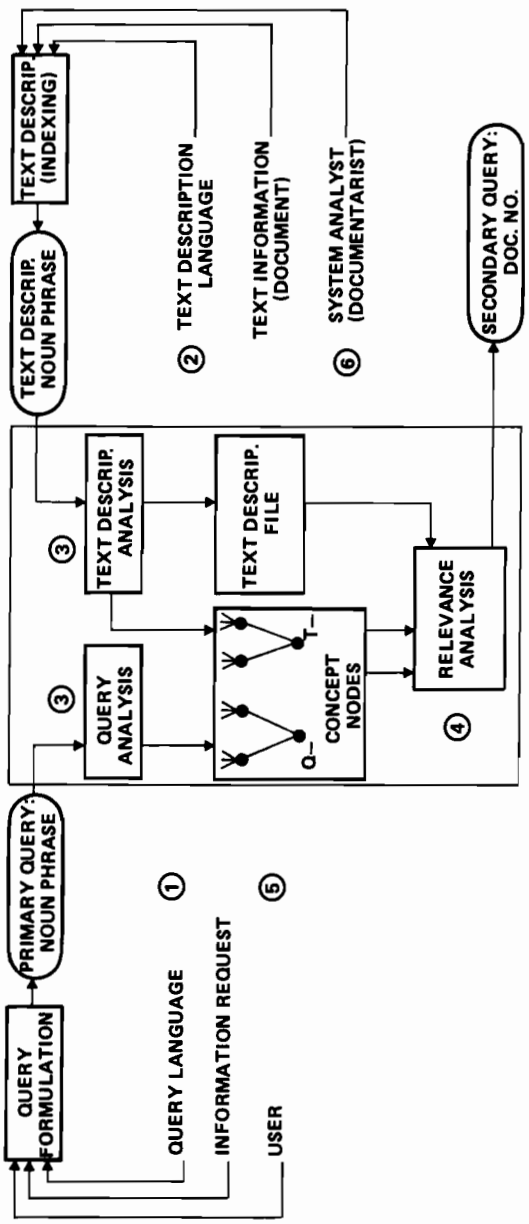


Figure 6. Information flow.

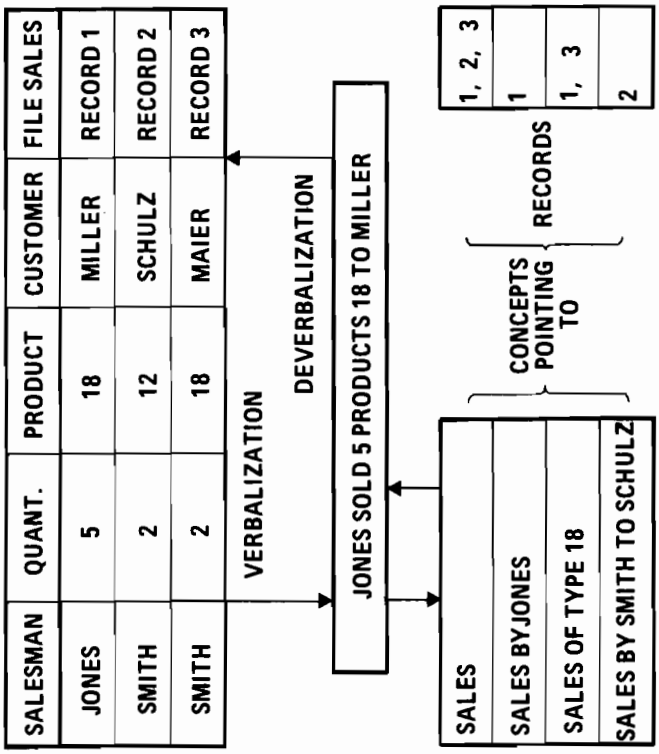


Figure 7. Application of noun phrase to formatted data.

REFERENCES

[1] Date, C.J., *An Introduction to Database Systems*, Addison-Wesley, Reading, Mass., 1975.

[2] Salton, G., *Dynamic Information and Library Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1975.

[3] Martin, H., and E. Parker, *A Feature Analysis of Interactive Retrieval Systems*, Stanford, 1974.

[4] Klawiter-Pommer, H.T., and W.D. Hoffmann, Übersicht über die für den Leistungs vergleich mehrerer Literatur-Datenbasen wichtigsten Parameter, *Nachr. Dok.*, 27, 3 (1976).

[5] Lustig, G., Probleme der Wörterbuchentwicklung für das automatische Indexing und Retrieval, *Nachr. Dok.*, 25, 2 (1974).

[6] Salton, G., A. Wong, and C.T. Yu, Automatic Indexing Using Term Discrimination and Term Precision Measurements, *Informat. Process. & Management*, 12, 43-51.

[7] Seelbach, D., *Computerlinguistik und Dokumentation*, Vlg. Dokumentation, Munich, 1975.

[8] *IBM Storage and Information Retrieval System/Virtual Storage (STAIRS/VS) Program Reference Manual*, SH12-5400-0, IBM, Stuttgart.

[9] Soergel, D., *Dokumentation und Organisation des Wissens*, Duncker & Hunblot, Berlin, 1971.

[10] Lang, F.H., Automatisierte Herstellung von Thesauren und Begriffssystemen für Wörterbücher und Fachterminologien, *Nachr. Dok.*, 24, 6 (1973) 231-238.

[11] Fillmoore, C.J., The Case for Case, in E. Bach and R.T. Harms, eds., *Universals in Linguistic Theory*, Holt, Rinehart and Winston, New York, 1968.

[12] Bruce, B., Case Systems for Natural Language, *Artificial Intelligence*, 6 (1975), 327-360.

[13] Lancaster, F.W., Problems of Communication in the Operation of Information Storage and Retrieval Systems, in J.S. Petöfi, and A. Podlech, *Fachsprache-Umgangssprache*, Scriptor, Kronberg, 1975.

[14] Sparck Jones, K., *Automatic Indexing 1974, A State-of-the-Art Review*, Computer Laboratory University of Cambridge, Cambridge, 1974.

Use of a Problem Solver for Data Base Handling*

E. Tyugu

An intelligent data base system is considered to contain the following three parts (see Figure 1):

- A linguistic (L) part for natural language processing;
- A large data base (DB) with efficient access methods;
- A data base handler that accepts questions and amendments as input, and prints out a program for data manipulation (DM).

If the interfaces between these parts are specified then it would be possible to design any of the three parts separately. There may then be some extra expense--for instance, the semantic support of parsing may be rather expensive in the linguistic part, if the other parts are used for this purpose.

We have designed a data base in which a problem solver has been used as the data base handler. The solver had initially been designed for general purpose use, and has been applied in different computer aided design systems [1], and redesigned for better efficiency later [2].

The source language of the solver is an extendible problem description language UTOPIST, which appeared quite convenient for representing semantics in the question-answering system.

The output of the solver is a program ready to run in an OS/360 environment. The interface between the solver and the data base is arranged on the data control language level and standard access methods of the operating system are used.

The main structure of the UTOPIST language is a declaration:

```
                    ⎧ primary-object
                    ⎪ dcl
dcl: = id,...:      ⎨ ([          ;...)
                    ⎪ relation
                    ⎩ notion [(amendment)]
```

*Report of an informal presentation.

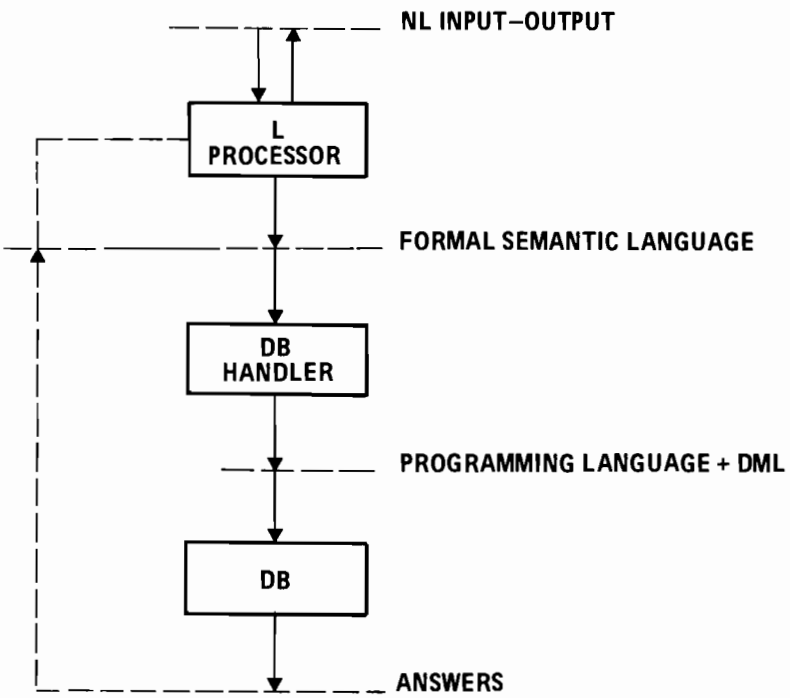

NL INPUT–OUTPUT
L
PROCESSOR
FORMAL SEMANTIC LANGUAGE
DB
HANDLER
PROGRAMMING LANGUAGE + DML
DB
ANSWERS

Figure 1.

An identifier that has been already used in a declaration denotes a notion and may be used for declaring new notions, and amendments may be made thereby. So, it is possible to declare:

```
PERSON:  (NAME, CHRN, PATR:  STRING:  AGE:  INT);
CHILD:  PERSON (PATR=CHRN of FATHER);
```

Relations may be expressed in different ways: by programs, equations, etc. One special way is representing them by relationships (tables) as in Codd's relational data base. An important point is that any relation is represented in such a way that it is possible to immediately apply it for calculation. (Any relation is formally regarded as a set of assignments and statements that may be used as soon as would be useful in solving a problem.)

Internal representation of the text in the solver is a semantic network (SN) with object and relations expressed in Figure 2 as nodes of different shapes. The following example is illustrated in Figure 2.

It is assumed that the notions:

FILE	CAR
FILTER	PERSON
PRINT	

are known to the solver, i.e. that their semantics are already represented by SN in the memory of the solver, and not in the data base. The notions in the left column are generally useful for data base problems. Those in the right column belong to particular problem areas.

The question "who are the owners of red cars?" will be in formal semantic language, as follows:

```
LET PROBL : (CARS, REDC : FILE(CAR);
    F1 : FILTER (FROM CARS TO REDC,
         COND = (COLOUR OF CAR OF CARS = 'RED'));
    WANTED : PRINT (OWNER OF CAR OF REDC FROM REDC));
PROG  CALCULATE WANTED FROM CARS;
END;
```

Using the semantic network as illustrated in Figure 2, the solver synthesizes an algorithm for solving the problem and generates a code that includes all required data description and data manipulation statements.

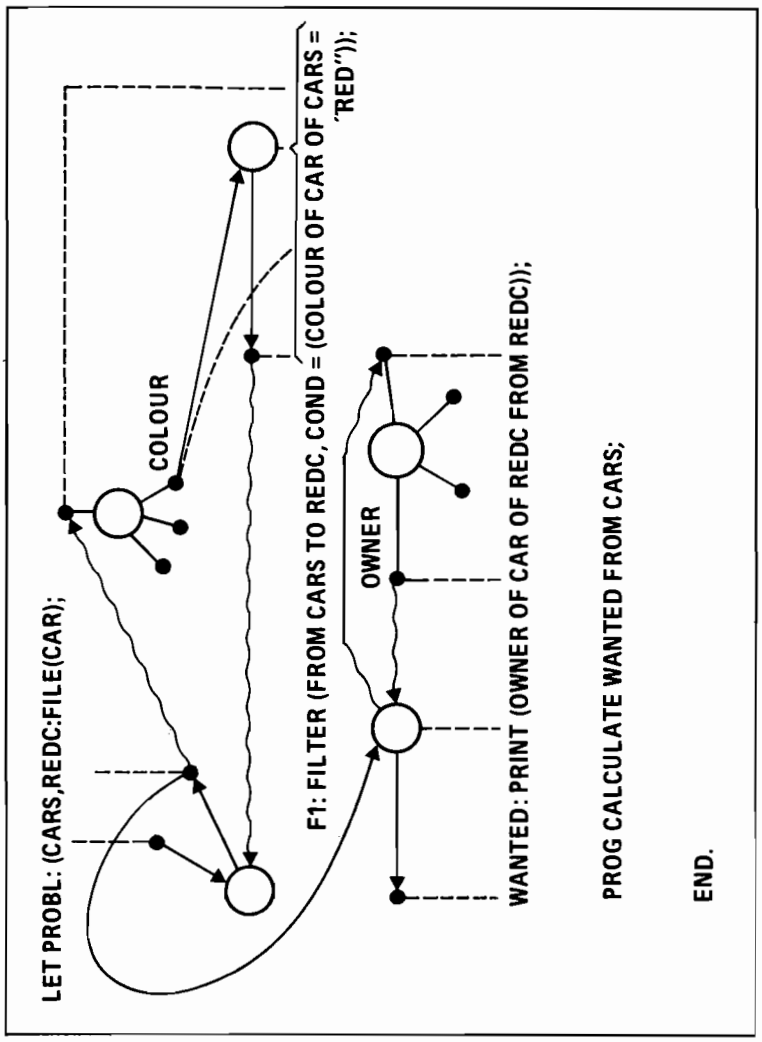

LET PROBL: (CARS,REDC:FILE(CAR);
COLOUR
F1: FILTER (FROM CARS TO REDC, COND = (COLOUR OF CAR OF CARS = 'RED'));
OWNER
WANTED: PRINT (OWNER OF CAR OF REDC FROM REDC));
PROG CALCULATE WANTED FROM CARS;
END.

Figure 2.

REFERENCES

[1] Tyugu, E.H., A Data Base and Problem Solver for Computer Aided Design, *Proceedings IFIP 1971*, North-Holland, Amsterdam, 1971.

[2] Kahro, M.I., et al., A Programming System PRIZ, *System Programming and Computer Software*, 2, 1 (1976).

QUESTIONS ON NATURAL LANGUAGE UNDERSTANDING AND INTERFACES

A Dictionary as a Data Base*

G. Guckler

INTRODUCTION

In our research project on the investigation of the semantic structure of natural languages via data processing (Semstruktur)** a vocabulary of about 17,000 words with 50,000 units of meaning is being tested to see whether they are characterized--as we presume--by a hierarchical network-like structure, the highest units of which are terms or concepts (so-called semantic universals) that cannot be further defined. To this end we are attempting to identify units and relationships of a possible semantic structure of natural languages on an empirical and a traditional lexicographic basis. We expect that by investigating dictionary definitions, which as "definientia" principally have the form "genus and differentia specifica" or function as equivalences of other expressions, a communicative network will be discovered the core of which is a system of concepts shared by several languages.

Thus, the object of our project is to make explicit the semantic relationships between words and the concepts they denote, a relationship implicitly existing in a dictionary. Thus, the definition

anmachen ... 3.2 Feuer~ anzünden

establishes a semantic relationship (of equivalence) between the definiendum (key word or keyword plus semantic context) and the definiens. When looking up the head word "anzünden" as definiendum one finds

anzünden ... 1 zum Brennen, Glühen, Leuchten bringen

This procedure can be continued. For each expression the user finds equivalences and explanations of a conceptual and linguistic

*This report is a summary of earlier reports by G. Wahrig, E. Müller, and G. Guckler [1-15].

**The research is sponsored by the Deutsche Forschungs-Gemeinschaft (DFG), Bonn-Bad Godesberg and headed by Professor G. Wahrig of the University of Mainz.

nature. This is the inherent semantic structure, contained in each monolingual dictionary, provided it works with definitions. The semantic description of expressions can be regarded as directed from expression to contents. The first point is to describe the preparation of the linguistic data taking the lexicographic structure of the dictionary entries as our starting point and arriving at a storage in a data base of the STAIRS type [16,17,18]. Later sections deal with the resulting linguistic analyses based on data processing.

PROJECT CORPUS MATERIAL

The starting point and basis of the linguistic analysis are monolingual German dictionary entries.* First we limited ourselves to the study of basic vocabulary, the selection of lexical entries being based on pragmatic-stylistic criteria. This means that we deal primarily with lexical units which at least for a part of their respective meanings do not have to be marked by an additional reference such as "colloquial", "poetic usage", "dialect", "technical term", etc., but have a clearly representational function, as Karl Bühler used the term [20].

Lexicographic Preparation of the Material

The structure of dictionary entries is organized according to the various meanings of a key word in connection with its relevant context. Each word is followed by several explanations of meaning, which are listed separately by medium faced index numbers, the semantically relevant context preceding the definition proper. (The explanation of meaning is followed by usage examples.) Possible relationships between single definitions of a key word, such as relative of sub- or coordination between two or several meanings are indicated by a hierarchical arrangement of index numbers. Thus, key word or index number, semantically relevant context, definition, and example of usage are the most important items of information. In addition there is an attempt to guarantee a complete and extensive description of the respective lexical entry by including references to grammatical, syntactic and pragmatic-stylistic properties and by indicating the relationship to other entries of the dictionary by cross-references (such as synonyms, antonyms, etc.)

Taking the key word "absolut" as our example we show the above mentioned formal structure of entry in our dictionary (Figure 1).

---

*The project corpus material is in preparation [19].

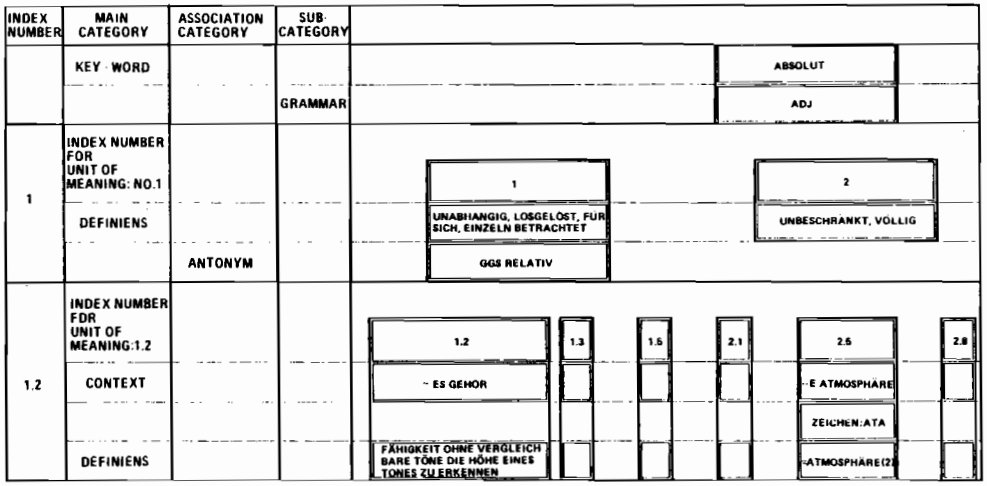


Figure 1.

ab so'lut Adj. 1 *unabhängig, losgelöst für sich, einzeln betrachtet, Ges. relativ* **1.1** ~e *Bewegung (physikalisch) nicht denkbar als ohne Bezugssystem* **1.2** ~es *Gehör, Fähigkeit, ohne vergleichbare Töne die Höhe eines Tones zu erkennen* **1.3** ~e *Feuchtigkeit Ggs. relative Feuchtigkeit der Luft ohne Rücksicht auf Temperatur* **1.4** ~e *Helligkeit H. eines Sternes, ungeachtet der durch seine Entfernung bedingten Sichtbarkeit am Himmel* **1.5** ~e *Mehrheit M. von mehr als 50%* **2** *unbeschränkt, völlig* **2.1** ~er *Superlativ* ~ *Ggs.* relativ **2.2** ~e *Monarchie* = *Absolutismus* **2.3** ~es *Vertrauen unbedingtes V. ohne jede Einschränkung* **2.4** ~es *Maßsystem System der gesetzlich festgelegten Maßeinheiten* **2.5** ~e *Atmosphäre (Zeichen ata)* = *Atmosphäre(2)* **2.6** ~er *Nullpunkt (theoretisch) tiefste erreichbare Temperatur (-273,16 °C)* **2.7** ~e *Temperatur auf den ~en Nullpunkt bezogene T.* **2.8** ~e *Zahl eine ohne Vorzeichen betrachtete Z.* **3** *unbedingt* **3.1** ~e *Kunst ungegenständl., abstrakte K.* **3.2** ~e *Musik M., der keine außermusikalischen Vorstellungen zugrunde liegen* **3.3** ~e *Rechte R., die gegenüber jedem wirksam sind* **3.4** ~er *Scheidungsgrund S., der ein Verlangen nach Scheidung rechtfertigt, ohne Rücksicht darauf, ob die Ehe zerrüttet ist* **3.5** *kein.* ~er *Alkohol wasserfreier Äthylalkohol* **4** *(adv.) durchaus, ganzlich, überhaupt, völlig das ist ~ unmöglich* [<lat. *absolutus, Part. Perf. zu absolvere* „lösen"]

Formal Preparation of Dictionary Entries for Data Processing

Structured Unit of Information

As can be seen from the dictionary entry "absolut", an entry can consist of more than one unit of meaning. Each unit of meaning together with the key word forms an independent unit of information that can contain all previously mentioned types of information including the additional references as to grammar, style, etc. This total unit of information in the following will be called "structured unit of information".

Categories within the Structured Unit of Information

According to the lexicographic preparation of the dictionary entry a structured unit of information has the following organization:

Main Categories

(0) Key word;
(1) Index number for semantic unit; } identical from the functional point of view

(2) Semantically relevant context;

(3) Definiens;

(4) Example.

Only the types of information 0 (or) 1 and 3 are obligatory, the other two are optional depending on the respective key word. These most important types of information within a structured unit of information will henceforth be called "main categories (0 - 4)". In addition to these further optional types of information--either of a subordinating or of an associating nature--can be present, concerning either the key word as a whole or parts of it:

Subcategories

(1) Syllabification;

(2) Pronunciation;

(3) Grammar;

(4) Level of speech;

(5) Technical terminology;

(6) Geographical distribution;

(7) Situational usage;

(8) Reference of the type: ⟨... stands for⟩, ⟨... of⟩, the lemma following in main category 3;

(9) Cross-references ⟨abbreviation: ...⟩, ⟨symbol: ...⟩, related to the key word in main category 0.

These can be added to all main categories (0 - 4), except for subcategories (1 - 3) which can only appear in connection with the main categories 0 or 1 and 2.

A final group of types of information within the system of categories is represented by the following cross-references to other key words:

Association Categories

(1) Orthographic variants;

(2) Orthographic variants of technical terms;

(3) Synonyms;

(4) Antonyms;

(5) References to a key word under which further examples of usage are listed;

(6) Reference to an entry that is a general or associated term, under which a group of words are defined as examples;

(7) Complementary cross-reference to 1 and 3 of this group.

This group can be combined with main category 3 or replace it.

Numerical Code for the Combination of All Three Groups of Categories

In a dictionary entry there are various forms of dependencies between the main categories, subcategories, and association categories. In order to be able to arrive at a formal combination of them we gave each category within a group an index number (see lists above). The three groups can thus be combined by three-digit number codes. The main categories are always to be found in the first digit of the numerical code. The association categories appear exclusively in connection with the main category 3 and are to be found in the second position. With main categories 0 or 1 and 2 the subcategories come as second digit, and with main categories 3 and 4 or as subcategories of a secondary degree (that is if they are preceded by another subcategory) they appear as third digit in the code. It applies to all categories that within a structured unit of information they have to appear in ascending sequence. If the main categories are not complemented by sub- or association categories, the tens and single-digit positions are filled with zeros.

Figure 2 shows all possible combinations of the system of categories. Figure 3 shows the organization of data as a system of categories.

GROUPS OF CATEGORIES			
MAIN CAT.	ASSOCIATION CAT.	SUB-CATEGORY OF 1st DEGREE	2nd DEGREE
HUNDREDS	TENS		SINGLES
CODE → NUMBER ↓	1 2 3 4 5 6 7	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
0 OR 1			
2			
3			
4			

Figure 2. Combinations of categories.

```
200     □e #Atmosphäre#
290     Zeichen: ata
370     Atmosphäre (2)
□
aaa     1500414
100     2.6
200     □er #Nullpunkt#
300     die tiefste erreichbare Temperatur (-273, 16§347C)
□
aaa     1500415
100     2.7
200     □e #Temperatur#
300     auf den □en Nullpunkt bezogene T.
□
aaa     1500416
100     2.8
200     □e #Zahl#
300     eine ohne Vorzeichen betrachtete Z.
□
aaa     1500417
100     3
300     unbedingt
□
aaa     1500418
100     3.1
200     □e #Kunst#
300     ungegenständl., abstrakte K.
□
aaa     1500419
100     3.2
200     □e #Musik#
300     ungegenständl. aqqq
300     M., der keine au§s39ermusikalischen Vorstellungen zugrunde
        liegen
□
aaa     1500420
100     3.3
200     □e #Rechte#
300     R., die gegenüber jedem wirksam sind
□
aaa     1500421
100     3.4
200     □er #Scheidungsgrund#
300     S., der ein Verlangen nach Scheidung rechtfertigt, ohne
        Rücksicht darauf, ob die Ehe zerrüttet ist
□
aaa     1500422
100     3.5
300     rein
□
aaa     1500423
100     3.5.1
200     □er #Alkohol#
300     wasserfreier Äthylalkohol
□
aaa     1500424
100     4
130     50
300     durchaus, gänzlich, überhaupt, völlig
400     das ist □ unmöglich
□
```

Figure 3.

```
aaa     1500401
000     ab.so'lut
030     Adj.
□
aaa     1500402
100     1
300     unabhängig, losgelöst, für sich, einzeln betrachtet
340     relativ
□
aaa     1500403
100     1.1
200     □ e #Bewegung#
300     (physikalisch nicht denkbare) B. ohne Bezugssystem
□
aaa     1500404
100     1.2
200     □ es #Gehör#
300     Fähigkeit, ohne vergleichbare Töne die Höhe eines Tones
        zu erkennen
□
aaa     1500405
100     1.3
200     □ e #Feuchtigkeit#
300     F.sgehalt der Luft ohne Rücksicht auf Temperatur
□
aaa     1500406
100     1.4
200     □ e #Helligkeit#
300     H. eines Sternes, ungeachtet der durch seine Entfernung
        bedingten Sichtbarkeit am Himmel
□
aaa     1500407
100     1.5
200     □ e #Mehrheit#
300     M. von mehr als 50 §364
□
aaa     1500408
100     2.
300     unbeschränkt, völlig
□
aaa     1500409
100     2.1
200     □ er #Superlativ#
370     Elativ
□
aaa     1500410
100     2.2
200     □ e #Monarchie#
370     Absolutismus
□
aaa     1500411
100     2.3
200     □ es #Vertrauen#
300     unbedingtes V. ohne jede Einschränkung
□
aaa     1500412
100     2.4
200     □ es #Ma§s39system#
300     System der gesetzlich festgelegten Ma§s39einheiten
□
aaa     1500413
100     2.5
```

Figure 3 (continued)

Preparation of the Dictionary for the Data Base

The formal organization of the dictionary entries has been conceived with regard to their storage in a data base, since we were fortunate enough to receive the cooperation of the Zentralstelle für Machinelle Dokumentation (ZMD) in Frankfurt. This institution working with the STAIRS system seemed to suit our purposes best and offered optimal conditions for the planned investigation.

The Assignment of the Individual Categories of Structured Units to Data Base Levels

The strict formal preparation of the dictionary entries provided the basis for their transfer into the structure of the above mentioned data base and made possible multiple forms of linguistic enquiry and the resulting information retrieval.

STAIRS levels within a STAIRS document: structure within the STAIRS levels is characterized by a hierarchical arrangement of levels. The search always refers to the individual levels. The document is the highest ranking unit within the data base. It is subdivided into STAIRS paragraphs which for their identification have paragraph codes and paragraph names. They are subdivided into sentences, separated by punctuation. The smallest unit within the document is the STAIRS word, delimited by blanks or punctuation marks.

The assignment of categories of structural information units to STAIRS-levels: in our attempt to adapt our system of categories to the STAIRS system we arrived at the following correspondence:

- A structured unit of information corresponds to a STAIRS document.
- Each of the main categories 1-4 corresponds to one STAIRS paragraph, whereas main category 0 corresponds to two paragraphs.
- The subcategories and association categories do not correspond to independent paragraphs but each of them forms a single sentence within these paragraphs.

A strict paralleling of all our categories with STAIRS paragraphs was not possible, since due to the multitude of combinations of main, sub- and association categories an unnecessarily complicated system would have arisen. Nevertheless, in order to provide access to all our categories of structured units of information via data base search we stored their three-digit number code together with the lexical text.

The three-digit number combination as a whole is of interest for retrieval of information as are individual digits within it. That is why the stored index number code was subdivided into hundreds, tens, and single digits, so that each individual position is accessible as a STAIRS word.

Figure 4 provided by the ZMD, again shows how our categories were integrated into the STAIRS system. As a further illustration we present an extract from a search query (Figure 5).

RETRIEVAL WITH STAIRS

Some Technical Aspects

In our discussion of technical aspects of retrieving with the STAIRS data base we confine ourselves to the possible uses to which we have put the system in our research program. Principally we can make use of all STAIRS functions--except for the SELECT function, which is related to formatted data only. As to the SEARCH function all types of search query statements can be formulated via operators, operands, and qualifiers. Furthermore, in addition to the qualifiers provided by STAIRS, our category number codes--although formally stored as words of contents--for our purposes function as qualifier and enable us to have better control of information retrieval.

Thus, taking into consideration further possibilities of combined search (operands, qualifiers, and operators), it follows that within our linguistic investigations we will hardly be able to exhaust the theoretical possibilities of the dictionary as a data base system.

Examples of Linguistic Investigations with the Data Base

Search Via Category Code

The search via the categories' code, which correspond to specific types of information in the dictionary, provides subsets of vocabulary clearly defined as to their function in the dictionary. According to the type of category there are the following subsets:

Main Categories

- The set of all keywords in the dictionary, of their syllables and--as a subset of all keywords--the set of all verbs with a separable prefix.

- The set of all keywords defined (at least for one of their meanings) within a semantically relevant context (MC 2).

- The set of all keywords for which there are given examples of usage in the dictionary (MC 4).

S T A I R S

Paragraph Name	Paragraph Code	STORAGE OF C O N T E N T S
BLCKNR	001	STRUCTURE: Code for type of dictionary entry: GGG = basic dictionary/German w. GFW = basic dictionary/foreign words GKO = basic dictionary/irregul. verbs blank six-digit number for documents/each dict. entry three-digit number/document n^{o} within a dict. entry: 000 resp. 001, 002 etc.
STICHW	010	- keyword (original)
	020	- keyword (only alphabetic text) for search query
	030	- keyword subdivided in stem and prefix (/)
	040	- keyword subdivided in syllables
		- Paragraphs 020, 040 are obligatory in the first document of a lexical entry; following documents contain paragraph 020 only.
SUBSTW	100	- Content of category with leading category code 0 (except: 000)
		- three-digit category codes
		- category codes subdivided in hundreds, tens and singles
SUBSTT	200	- Content of category with leading category code 1 (except: 100) storage of category codes as above
KONTEX	300	- Content of category with leading category code 2 storage of category codes as above
DEFINS	350	- Content of category with leading category code 3 storage of category codes as above
BEISPL	400	- Content of category with leading category code 4 storage of category codes as above

Figure 4. Structure of the data base.

DW0275004957

BLCKNR: GGG 102733 002

STICHW: Auge SST: 1

DEFINS: H3 Z0 E0 K300 Sinnesorgan des Menschen u. der Tiere zur Wahrnehmung von Lichtwellen.

BEISPL: H4 Z0 E0 K400 blaue, braune, graue *n; ein künstliches *; die*n; öffnen, aufschlagen, niederschlagen; mit den *n rollen; einen Fremdkörper im * haben; mir ist etwas ins * gekommen, geflogen; die *n brennen, tränen mir; jmdm. ein * ausschlagen; mit den *n zwinkern.

DW0275007221

BLCKNR: GGG 102944 002

STICHW: Begriff STT: 1

DEFINS: H3 Z0 E0 K30 meist mit einem Wort benannte, von den sinnlichen Empfindungen u. Wahrnehmungen abgeleitete Verallgemeinerung des Inhalts von Sachen der Außenwelt u. der menschlichen Vorstellung.

DW0275007328

BLCKNR: GGG 103348 012

STICHW: bei STT: 1.8.1

KONTEX: H2 Z0 E0 K200 die Gelegenheit *m Schopfe fassen. H 2 Z4 E0 K240 fig..

DEFINS: H3 Z0 E0 K300 eine günstige Gelegenheit schnell wahrnehmen.

DW0275013319

BLCKNR: GFW 115511 002

STICHW: Echo STT: 1

DEFINS: H3 Z0 E0 K300 reflektierte Schallwellen, die an ihrem Ausgangspunkt wieder wahrgenommen werden. H3Z3 E0 K330 Widerhall (1).

BEISPL: H4 Z0 30 K400 ein einfaches, dreifaches, mehrfaches *.

STICHW: empfinden STT: 1

KONTEX: H2 Z0 E0 K200 Reize *.

DEFINS: H3 Z0 E0 K300 mit den Sinnen wahrnehmen.

BEISPL: H4 Z0 E0 K400 Durst, Hunger, Kälte, Wärme *.

DW0275016093

BLCKNR: GGG 103567 002

STICHW: Erscheinung STT: 1

DEFINS: H3 Z0 E0 K300 wahrnehmbarer Vorgang.

BEISPL: H4 Z0 E0 K400 es ist eine auffallende, eigentümliche, seltene *; es ist eine bekannte *, daß ...; das ist eine typische * für ...

DW0275018770

BLCKNR: GGG 106659 003

STICHW: Freibeuter STT: 2

SUBSTT: H1 Z4 E0 K140 fig.; abwertend.

DEFINS: H3 Z0 E0 K300 jmd., der ohne Rücksicht auf Sitte u. Gesetz seinen Vorteil wahrnimmt.

Figure 5.

Association Categories

- The set of all keywords for which (at least for one of their meanings) there is a synonym, an antonym, or an orthographic variant (AC 1-4).

- The set of all keywords having a cross reference to other keywords with which they are in a semantically relevant relation (AC 5 and 6).

Subcategories

- The set of all keywords with additional information concerning syllabification (SC 1).

- The set of all keywords for which an explication of the pronunciation is necessary (e.g. foreign words, words with more than one possible pronunciation or accentuation) (SC 2).

- The set of all units of meaning having a stylistically or pragmatically marked dictionary entry (SC 4).

- The set of all units of meaning having a terminologically marked dictionary entry (SC 5).

- The set of all keywords having a geographically marked dictionary entry (SC 6).

- The set of all units of meaning confined in their usage to a special situation, generally marked as temporal or local (SC 7).

- The set of all keywords having a cross reference to other keywords with which they are in a formal relation (SC 8 and 9).

Search Via Contents of Category Code

Retrieval of elements stemming from the contents of the categories is possible only if the researcher knows which elements he wants to investigate. This is especially true for the systematically conventionalized aspect of descriptive language (metalanguage) in a dictionary. This is valid for the contents of several subcategories. Retrieval of this kind makes sense only in combination with category code. The result will be more specialized subsets:

Subcategory 1

- The set of all keywords at the syllabification of which there is a change from -ck- to -k. k-, or from -ll- to -ll.l-, etc.

Subcategory 2

- The contents of this category--which contains information on pronunciation and accentuation--depends on the corresponding keywords and thus are not conventionalized search operands.

Subcategory 3

- All subsets of the vocabulary each of which belongs to a special word class (nouns, verbs, adjectives, prepositions, conjunctions, adverbs, pronouns, prefixes, etc.).

Nouns

- The subsets of all nouns in the vocabulary having masculine or feminine or neuter gender.

- All subsets of the nouns that have a special ending for the genitive case and/or the plural.

- The subset of nouns for which the plural form is marked by the "umlaut".

- The subset of nouns that have no plural forms.

- The subset of nouns that have no singular forms.

Verbs

- The subsets of all verbs in the dictionary with strong or weak conjugation.

- The subsets of verbs being conjugated with "haben" or "sein".

- The subsets of the verbs that have a special syntactic "valency". (e.g. the set of all verbs with direct object, the set of all verbs with an obligatory time adverbial, the set of all verbs that may or must be reflexive, etc.)

Adjectives

- The subsets of adjectives that have a special syntactic "valency".

Subcategory 4

- The subsets of the set of all--at least for one of their meanings--stylistically marked keywords characterized by one of the following labels: fig. (= figürlich), geh. (= gehoben), scherzh. (= scherzhaft), umg. (= umgangssprachlich), veralt. (= veraltet), etc.

Subcategory 5

- The subsets of the set of all--at least for one of their meanings--terminologically marked keywords that belong to one of the specialized technical terminologies (e.g. the set of keywords that make part of the medical terminology).

Subcategory 6

- The subsets of all geographically marked keywords that each belong to one geographical part of the whole area of German language use (e.g. all keywords that in a special context or with a special meaning are used only in Austria).

Subcategory 7

- The subsets of all keywords that occur only in a restricted temporal or local environment; a type list of the contents of this category is in preparation and will provide for specific subsets.

Subcategory 8

- The subsets of all keywords that are word forms, abbreviations, symbols, or acronyms and that are referred to corresponding keywords.

Subcategory 9

- The subsets of all keywords for which there exists a generally known abbreviation, symbol, or acronym, or which can be represented by a numeral.

Search Via Elements of Natural Language

Finally there is the possibility of operating via operands that are part of natural speech and that are nonconventionalized parts of the semantic descriptive language. Such search operations are especially interesting for investigation of the elements of definientia (MC 300), which--as mentioned above--aims at the identification of semantic universals. By taking as search query statement an element of natural language the semantic relevance of which is intuitively assumed, the result will be the subset of keywords for the definition of which this element (concept, notion) is used. For this kind of research there are still the following problems: the elements used for the definitions can be polysemes or homographs; one concept can be realized by different synonymic words within the definientia, and furthermore the research operation is rendered more difficult by the fact that the search element is taken in its uninflected form (lemma) whereas the elements in the definitions are often supplied with

flexional morphemes, etc. Thus no complete or systematic search (result) can be expected, but rather a series of chance results. This is exactly where our work with the data base finds its limitation.

By giving the above list of research operations and retrieval results we could only indicate very incompletely the abundance of possible computational linguistic investigations given by the organization of our dictionary as a data base. By any combination of search operands (category codes and contents of categories) via Boole-operators we can set up more and more strict conditions so that we arrive at more and more special subsets of documents.

Linguistic Studies Beyond Data Base Use

In terms of linguistics our investigations are aimed at the construction of a hierarchy of mentally related terms in the category of definientia (main category 3). We are trying to find out which words--and corresponding terms or concepts--are used again and again in a dictionary for the semantic description of the units of a language. We think that within the set of these words we shall find those that constitute a kind of kernel for a content structure of a language.

This is why we are at present engaged in analyzing the words contained in our main category 300. Our data base can furnish us with the whole set of the material content of this category. For further investigations we have developed individual programs to generate indices, concordances, lists of tokens and types, lists of associated terms (synonyms, antonyms and related terms) for linguistic analyses being performed by computer or manually, depending on the aims and structures of the individual tasks.

The result of our project is expected to be a metalanguage for the semantic description of a natural language. This metalanguage will contain only semantically disambiguated words (terms) on a conventionalized basis and will correspond to the metalanguage for grammatical and pragmatical descriptions of dictionary entries contained in the subcategories of our data base.

At present a group of graduate students are engaged in pilot studies investigating the so-called "grammatical words" of the dictionary (data base) to discover the semantic deep structure of grammatical relations expressed by prepositions and adverbs. From the first outlines of their findings we are confident that in the near future we will have at our disposal a set of "relational primitives" for semantically describing the relations between the words (terms) within the definientiae and between definienda and definientiae. This set of relations will also be of interest for the further developement of data base systems for information retrieval.

In most data base systems in use today, descriptors in information retrieval systems are connected by logical operators that fundamentally function only as negation, conjunction, and disjunction. More sophisticated operators have been installed for special purposes (medicine, chemistry, jurisprudence) functioning as causative, resultative, and similar relations. In our system we can install a complete set of operators that function as relations representing the contents of a natural language grammar.

REFERENCES

[1] Guckler, G., E. Müller, and G. Wahrig, *Untersuchungen zur semantischen Struktur natürlicher Sprachen mittels EDV (Semstruktur), Zwischenbericht über die Arbeiten von Sept. 1974 bis Jan. 1976 am DFG-Projekt Wa 340/1*, DFG, Bonn-Bad Godesberg, and University of Mainz, Mainz, 1976.

[2] Müller, B., *Onomasiologie der deutschen Präpositionalphrasen mit lokalbezug*, Staatsexamensarbeit, Mainz, 1976.

[3] Wahrig, G., Informationsklassen in Wörterbüchern, Beitrag zum Kolloquium "Kasus-Labels, mehrsortige logische Konstrukte und handlungstheoretische Elemente in der Grammatik", Bielefeld, April, 1975 (mimeo).

[4] Wahrig, G., *Der Possesiv als Kategorie einer semantischen Tiefenstruktur der deutschen Sprache*, University of Mainz, Mainz 1976 (in preparation for *Festschrift für W. Th. Elwert*, 1977).

[5] Wahrig, G., *Untersuchungen zur semantischen Struktur natürlicher Sprachen (Semstruktur), Beschreibung des Projektes DFG Wa 340/1*, University of Mainz, Mainz, 1974.

[6] Wahrig, G., *Strukturelle Grammatik: Inhalte grammatischer Kategorien*, Vorlesungsskriptum, University of Mainz, Mainz, 1976.

[7] Wahrig, G., *Zwischenbericht über das DFG-Projekt Wa 340/1, Nachtrag*, University of Mainz, Mainz, August 1976.

[8] Wahrig, G., E. Müller, und G. Guckler, *Untersuchungen zur semantischen Struktur natürlicher Sprachen mittels EDV (Semstruktur), Zwischenbericht über die Arbeiten von Sept. 1974 bis Jan. 1976 am DFG-Projekt Wa 340/1*, DFG, Bonn-Bad Godesberg, and University of Mainz, Mainz, 1976.

[9] Wirth, I., *Funktionen deutscher Präpositionen mit Ausnahme der Lokal- und Temporalrelationen*, Staatsexamensarbeit, Mainz, 1977.

[10] Christmann, R., *Das Wortfeld für "Zeit" im Deutschen*, Staatsexamensarbeit, Mainz, 1977.

[11] Gotthardt, W., Bericht über eine graphentheoretische Untersuchung zur Beschreibung der semantischen Struktur des Grundwortschatzes, in *Zwischenbericht über das DFG-Projekt Wa 340/1*, Nachtrag, DFG, Bonn-Bad Godesberg, and University of Mainz, Mainz, August 1976.

[12] *Form und Funktion von Definitionen in Wörterbüchern*, Mag.-Arbeit, Lessenich-Drucklieb, Mainz, 1976.

[13] Dahlberg, I., Über Gegenstände, Begriffe, Definitionen und Benennungen, in *Muttersprache 2*, Wiesbaden, 1976.

[14] Guckler, *Zweisprachiges Wörterbuch für angenäherte operationelle Analyse semantischer Entsprecherungen mittels EDV*, Niemeyer, Tübingen, 1975.

[15] Wahrig, G., *Anleitung zur grammatisch-semantischen Beschreibung lexikalischer Einheiten, Versuch eines Modells*, Niemeyer, Tübingen, 1973.

[16] Lizenzprogramme, *Übersicht über das Storage und Information Retrieval System (System zur Informations-Speicherung und Wiedergewinnung) für die Systems IBM/360 und IBM/370 (OS)*, IBM, Stuttgart, 1974.

[17] IBM, *STAIRS-Handbuch, Teil A: Funktionen und Möglichkeiten*, and *Teil B: Von der Datenerfassung zum betreibsbereiten System*, IBM, Stuttgart, 1973.

[18] *Zusatzprogramme für den Einsatz von STAIRS im I & D-Bereich, ZMD-Schrift*, IBM, Stuttgart, 1974.

[19] Wahrig, G., ed., *Der deutsche Grundwortschatz, Stuktur und Bedeutung*, Deutscher Taschenbuch Verlag, Munich (in preparation).

[20] Bühler, K., *Sprachtheorie, Die Darstellungsfunktion der Sprache*, Fischer, Stuttgart, 1965.

The Role of Prepositions in Understanding Relations of Causality

G. Lau

INTRODUCTION

Recently the field of causality and motivation has gained considerable interest. The use of concepts of causality has become important in artificial intelligence (AI) research and in the theory of argumentation. "How do we answer why-questions?" This problem has become central for AI. On the other hand in the theory of argumentation the main problem consists in finding a way for the generation of answers to questions like "How can you know that?". AI--in other words--is more bound to the object level of communication, while the theory of argumentation deals with a meta level. But pure separation of object and meta level seems to be impossible, as the psychologist Piaget has shown us. (He distinguishes between three different kinds of explanations, causal (the connection between cause and effect), logical (connections between propositions), and psychological (a mixture--reasons and actions)[1]).

When considering natural language (NL) interaction with computer data bases, both levels surely have relevance. A user should be able to get an answer to his question, why some event, of which he has been told by the system, has come into existence. There will be situations when the user has some doubts with respect to information coming out of the system. He wants to know how the system has formed that information. In a computer system a number of preconditions must be simulated significant for verbal communications between humans [2]. The following is my free translation of [2].

SOME BASIC REQUIREMENTS FOR ANSWERING QUESTIONS

- Everybody who starts talking needs one or more individuals to talk to.
- The individuals involved in verbal communication agree in basic ideas how to act, i.e. they have a definition of the situation in which they communicate. The possibility of cooperating in that situation is founded on that definition.
- The situation is a social one, in so far as the involved individuals react with each other.

- One of the individuals needs some further definition of the situation.

It is a topical question and it calls for a widening of the definition of the situation. The question can be directed backwards: then a presupposed part of the definition is being challenged. Or the question can be directed forwards: the questioner asks for some more detailed specification of the situation.

A minimal cycle of communication consists of the following triple:

D_i, the set of presupposed utterances that consitutes the definition of the situation up to the moment of the question;

F_j, the question presupposing D_i; and

D_k, the utterance that is an answer to question F_j and gives a specification of D_i.

This triple of subsequent utterances serves as a scheme for answering questions put explicitly, as well as a basis for understanding texts. When understanding texts we have the task of finding questions F_j by which we could have asked for the information given in D_k on the basis of D_i.

Let us consider the following quite different sequences of utterances as an example:

	I	II	III
D_i	... The recorder is switched on.	....	... The recorder is switched on.
F_j	And why?	Why is the recorder switched on?	....
D_k	I have pressed the power switch.	I have pressed the power switch.	I have pressed the power switch.

The dots in front of the information in D_i are to indicate that a complete definition of the situation would be more than the sentence in D_i. That will be understood at once when we consider the question F_j in sequence I: the why-question is ambiguous. At least in German, you can ask for the goal as well as for a cause when using "warum?". In order to be able to answer

F_j in I, another part of D_i has to be presupposed: the questioner is more likely to need information about the cause of the focused element of the situation, than to need information about its goal.

In I and II the question F_j is being put explicitly. But in III, which serves as an example for a text, such a question would be quite unnatural. The hearer must decide whether he supposes a causal relation exists between the two facts. The result of his understanding is only equal to I and II if he assumes an F_j as in I to be natural.

DIFFERENT KINDS OF CAUSALITY

There is no standard subdivision of the different kinds of causality. Those that have been introduced in AI differ in several aspects [cf. 3]: there is a different number of concepts; and the same concepts are used to combine units of language which are of quite different complexity. Schank [4] for instance uses a reductionistic way of representing the meaning of NL inputs. His four causal relations (ENABLE, RESULT, REASON, INITIATE) are defined to combine only basic conceptualizations. Rumelhart [5] does not say much about the complexity of the events, actions, etc. which are the facts of the relations of ALLOW, INITIATE, MOTIVATE, or CAUSE. Definitions of causal relations have been very vague so far in AI. It is useful to study the results of logicians and philosophers [6,7]. In their work we find exact definitions of causal concepts; and we need exact definitions as soon as we deal with inferences.

The lack of a standardized subdivision of different kinds of causality encouraged me to collect causal words from the German language in order to have enough material for an extensive analysis. In this paper I want to deal with prepositions only. But that does not mean that prepositions are more important than causal verbs (VERURSACHEN, FOLGERN, etc.), nouns (URSACHE, GRUND, MOTIV, etc.) and small words like conjunctions (DENN, WEIL, etc.) and particles (NÄMLICH, FOLGLICH, JA, etc.) which play a very important role.

SOME CHARACTERISTICS OF PREPOSITIONAL NOUN PHRASES

The specific characteristics of prepositional noun phrases need a twofold answer: one answer reflects the problems of the generation of utterances; and the other deals with problems of understanding a given noun phrase.

The Generation of Prepositional Noun Phrases

When a communicator C_1 wants to tell his partner C_2 of the existence of some causal relation between two facts, he may build

a prepositional noun phrase and embed it in a sentence. This prepositional noun phrase in German may appear at different positions in the sentence. It is part of the sentence. The prepositional part of the noun phrase puts the nominal part (together with its attributes) into relation with the rest of the sentence. It often happens that a noun phrase does not express all the details that would be expressed by a whole sentence. Noun phrases sometimes demand ellipses in order not to be too long.*

DER RECORDER IST INFOLGE DES NIEDERDRÜCKENS VON POWER (DURCH MICH) EINGESCHALTET.

(The part in brackets may be left out.)

In this example, C_1 has to make decisions about the completeness of his verbalization: he may or may not leave out the agent. To a certain extent his decision depends on vocabulary. (He may for instance use TASTENDRUCK instead of NIEDERDRÜCKEN VON POWER.) But his decision to leave out this or that part also depends on the choice of prepositions: elements of meaning may move from a nominal part of the noun phrase to the prepositional part, as you can see from the following examples:

DER RECORDER IST WEGEN DES PLANS, PETER ZU ERFREUEN, EINGESCHALTET.
DER RECORDER IST ZU PETERS SPASS EINGESCHALTET.
DER RECORDER IST PETER ZULIEBE EINGESCHALTET.

We cannot say that the last sentence is elliptical as compared with the first one. We just find a short expression containing all the information of the longer one. It is the use of a certain preposition (ZULIEBE) that makes it possible to save words without losing information. Thus the generation and the form of causal prepositional noun phrases depends on the meaning of prepositions. A language having few causal prepositions offers only a few possibilities for leaving out nominal parts without loss of information.

Understanding Causal Prepositional Phrases

A well defined formulation of the meaning to be verbalized was the starting point for our considerations of how to generate noun phrases. Now, in the process of the rational reconstruction

*I must write most of the examples in German now, although I know this will make the paper more disagreeable.

of the understanding of noun phrases, a formulation in NL is our starting point. The formulation needs translation into some representation of meaning. When thinking of generating noun phrases, it was possible to restrict the field of relations to be handled. If, for instance, we disallowed the meaning "meeting somebody's interests", nothing would be wrong. ZULIEBE would never be used.

In simulating the process of understanding, things are reversed: no meaning may be disallowed, for the hearer C_2 cannot hinder his partner C_1 in wanting to convey some specific meaning. Therefore it is the task of C_2 to try to understand everything C_1 says. In order to be able to do that, he needs procedures for translating all words of his language into some other representation. The simulation of understanding processes drives us to the systematization of vocabulary. In doing this, we become conscious of the differences in the meaning of words. There is a result of this activity relevant for the generation of prepositional noun phrases: the inventory of concepts in our well defined area increases as soon as we start systematical studies in lexicology.

When we want to solve some of the problems of the generation of prepositional noun phrases, we should undertake two subsequent steps--collect all causal prepositions of some language, and judge their differentiation of meaning.

Inventory of Causal Prepositions in the German Language

I. Prepositions expressing causality but nothing else:

ANGESICHTS	LAUT
ANHAND	MANGELS
ANLÄSSLICH	MITTELS(T)
AUFGRUND	RÜCKSICHTLICH
BEHUFS	UM ... WILLEN
DANK	VERMÖGE
EINGEDENK	VON ... WEGEN
ENTSPRECHEND	WEGEN
GEMÄSS	ZUFOLGE
HALBER	ZULIEBE
HINBLICKLICH	ZWECKS
KRAFT	

II. Prepositions expressing causality or something else:

AUF ... HIN	(OB)
AUS	ÜBER
BEI	UM
DURCH	UNTER
IN	VON
MIT	FÜR
NACH	VOR (LAUTER)
ZU	

III. Prepositions dependent on certain verbs, nouns, and adjectives that constitute a causal relation between the predicate and an object:

ER LEIDET AN EINEM HERZFEHLER.
SIE IST STOLZ AUF IHRE PRÜFUNG.
ÜBER DIESEN SPASS GAB ES VIEL GELÄCHTER.

Group I consists of 23 prepositions, and group II of 15. Group III prepositions can only express causality when they are in connection with other words: they lack any proper causal meaning themselves and are ignored in the following. Of the 38 prepositions in groups I and II, what features can we find to build a subcategorization?

Let us consider the following hypothesis that I want to use during the analysis of prepositions:

- Precondition: Two prepositions are under consideration. PREP1 has an extension such that it is wider than the extension of PREP2 and includes the extension of PREP2. Here we usually say that the intension of PREP2 is greater than the intension of PREP1. (This would be the case for instance with PREP1 = WEGEN and PREP2 = ZULIEBE.)

- Hypothesis: The preposition with the greater intension has historically come into existence because it summarizes a relatively complex causal relation that has to be communicated frequently.

Prepositions with a great intension reflect the specific features of causal relations essential for the praxis of some society. This assumption may be thought of as having some relevance for problems that arise in understanding NL. Let us take a verbal expression that carries information about causality in an incomplete or inaccurate way. If C_2 needs more exact information about detailed causal features, for instance in order to be able to go on acting, then he needs heuristics that tell him how to complete his information. It seems quite natural that his subjective heuristics accord with specific intensions proved to be important for his society in so far as they are carried by prepositions. The individual then could be thought of as following exactly those heuristics that have got objective significance, in so far as they have been put into the verbal form of prepositions. The preposition is the outward form of certain heuristics to gather information about causality. We learn heuristics for language understanding in the field of causality by learning the meaning of prepositions, which is their essence.

On the basis of this hypothesis our analysis of the meaning of prepositions will have importance far beyond the field of understanding prepositional noun phrases. The combination of

causal features of prepositions would serve as patterns of the activity of understanding, even when no prepositional phrase appears. This is the case when somebody tries to complete his information, or tries to find a more detailed specification of causality than he could find explictly in some verbal expression.

In this paper I want to undertake only the first step towards a verification of the hypothesis: I want to find a plausible subcategorization of causal prepositions, and I want to show its use during the process of understanding causal prepositional phrases that are elliptical.

SOME SYNTACTICAL PECULIARITIES OF GERMAN CAUSAL PREPOSITIONS

The word *pre*position is misguiding, for German prepositions do not always appear in front of the noun phrase (NP). The following possibilities for the position of prepositions can be found:

- Position always in front of the NP, e.g. MITTELS;
- Position in front of or at the end of the NP, e.g. WEGEN;
- Two-part preposition with one part in front of, and the other part at the end of the NP, e.g. UM ... WILLEN;
- Position always at the end of the NP, e.g. HALBER;
- Preposition consisting of several parts in front of the NP, e.g. VOR LAUTER.

The second case sometimes carries ambiguity:

> PETER ENTSINNT SICH DES RECORDERS WEGEN DER TONBANDAUFNAHME.

This sentence can be understood in two different ways:

- DER ANBLICK DES RECORDERS ERINNERT PETER DARAN, DASS ER EINE TONBANDAUFNAHME MACHEN WOLLTE. (WEGEN is being considered as a preposition at the end of the NP.)
- PETER ENTSINNT SICH DES RECORDERS, DER FÜR DIE TONBAND-AUFNAHME BENÖTIGT WIRD. (WEGEN is being considered as a preposition in front of the NP.)

Sometimes in German a preposition and an article, connected with one another, form one word. We must distinguish between two cases:

- Indissoluble fusion, which needs not be analyzed; these prepositions are special words, e.g. PETER IST BEIM AUFNEHMEN.

- Fusion should be analyzed, e.g. PETER DRÜCKT ZUM STARTEN DES MOTORS AUF POWER.

One more peculiarity of prepositions that appear at the end of the NP is that they sometimes are separated by attributes from the NP that they really connect syntactically with the rest of the sentence:

DER KONTROLLAMPE AM OBEREN RAND DES RECORDERS NACH MÜSSTE DER RECORDER EINGESCHALTET SEIN.

Morphosyntax of German NPs can be considered as being solved to a high degree. There exist algorithms for an analysis that goes far beyond the complexity of NPs of the sort being used in contemporary AI systems [8]. In German some problems that in English require semantic analysis can be solved by using syntax analysis. But this is not the place to restart the debate whether syntax or semantics are more fundamental in writing algorithms.

SEMANTIC SUBCATEGORIZATION OF CAUSAL PREPOSITIONS

Three questions were put in order to find a first, tentative subcategorization: what is the specific quality of the first state/event expressed by the sentence? What is the specific quality of the second state/event expressed by the NP? What is the quality of the relation between them? An example for GEMÄSS:

PETER DRUCKT GEMÄSS PUNKT 3 DER BETRIEBSANLEITUNG AUF POWER.

- First state/event: PETER DRÜCKT AUF POWER--classification as an action;
- Second state/event: PUNKT 3 DER BETRIEBSANLEITUNG SCHREIBT ETWAS VOR--classification as a technical norm;
- Quality of the relation: an action is guided by a technical norm.

In Table 1 the semicolon divides each group into those prepositions that carry only causal meaning and those that might carry other meanings as well (local, temporal, etc.). Note that the members of one group cannot at will be replaced by some other member of the same group for two reasons. The causal relations have not been subcategorized according to every single difference in their quality in order to get a limited number of groups; thus prepositions rare in use were neglected a little. The subcategorization does not contain information on what sort of ellipsis

Table 1. Preposition groups.

- The first event is in the sense of physical causation brought about by the second event: INFOLGE, ZUFOLGE, WEGEN; DURCH, VON.
- The second state/event serves as a symptom of the first state/event: NACH (at the end of the NP).
- The first state/event is an action that has been initiated by the second state/event: AUF ... HIN.
- The first state/event is some conduct depending on the second state/event which is some sensation of the same individual: VOR LAUTER; AUS, IN, VOR.
- The first state/event is an action guided by the second state/event being some rule, convention, law, prescription: ANGESICHTS, ANHAND, AUFGRUND, EINGEDENK, ENTSPRECHEND, GEMÄSS, LAUT, VON ... WEGEN; NACH.
- The first state/event is an action, and the acting individual tries to bring about the second state/event: BEHUFS, HALBER, ZWECKS, HINBLICKLICH, EINGEDENK, ZULIEBE, RÜCKSICHTLICH, UM ... WILLEN, WEGEN, UM, FÜR.
- The first state/event is some conduct that may be interpreted as the second state/event which is an action: ANLÄSSLICH, ZU.
- The first state/event can only come about by the second state/event which is a continuous precondition: DANK, KRAFT, VERMÖGE, MANGELS; ÜBER, BEI, UNTER, VOR LAUTER.
- The first state/event is the opposite of some state/event, for which the second state/event is a nonfulfilled precondition: MANGELS.
(Prepositions indicating the use of tools and means are omitted.)

is possible for the second state/event, expressed by the NP. For instance ZWECKS and ZULIEBE are members of the same group, but it can easily be shown that they cannot be used synonymously:

PETER SCHALTET DEN RECORDER EIN, DAMIT PAUL SICH FREUT.

This sentence can be paraphrased using NPs with ZWECKS or ZULIEBE:

PETER SCHALTET PAUL ZULIEBE DEN RECORDER EIN. (Not possible: ZWECKS PAULS.)
PETER SCHALTET ZWECKS DER ERMUNTERUNG PAULS DEN RECORDER EIN. (Not possible: DER ERMUNTERUNG PAULS ZULIEBE.)

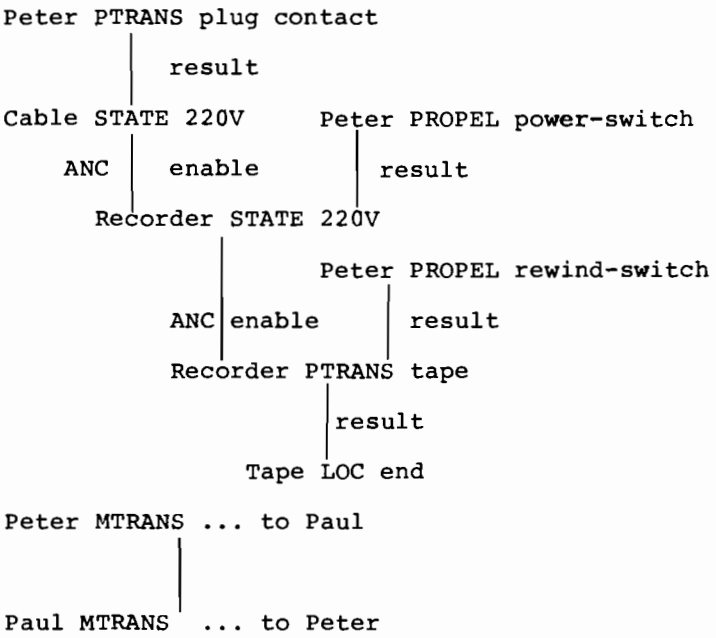
THE USE OF THE SUBCATEGORIZATION IN HEURISTICS FOR UNDERSTANDING NPs

First we shall have to establish a definition of a situation. I will use two ways of illustrating details of the definition: the reductionistic representation of Schank [9], and the more complex diagrams of Goldman [7, p. 30ff.]. I hope they are familiar to the reader.

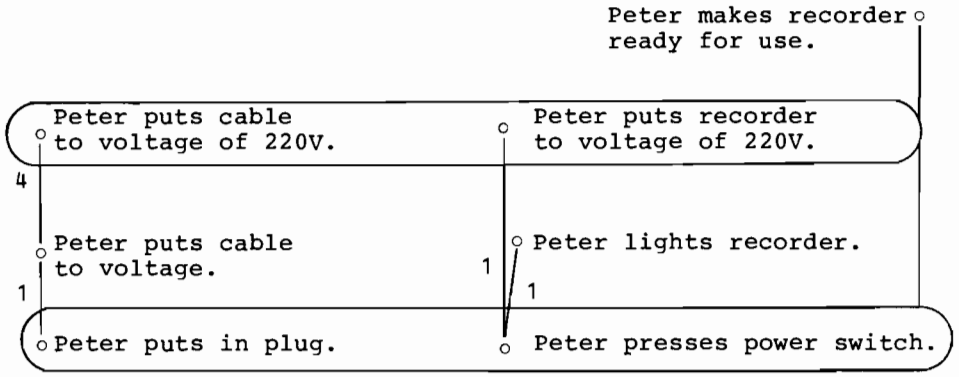
The example of a story will give us an idea of the situation we are talking about. The specific utterance will be translated into Schank's and Goldman's diagrams. After that we will continue the story with a question that will be answered by four alternative NPs, differing only in the prepositions. We will show how the information underlying our subcategorization can guide the understanding of the NPs.

Text PETER UND PAUL MACHEN EINE TONBANDAUFNAHME. PETER STECKT DAS KABEL EIN, DRÜCKT DIE POWER-TASTE, UND SPULT ZURÜCK, UM DEN RECORDER FÜR DIE AUFNAHME BEREITZUMACHEN. DANN VEREINBAREN PETER UND PAUL, DASS PAUL ZU SPRECHEN BEGINNEN SOLL, SOBALD PETER DIE AUFNAHME STARTET.

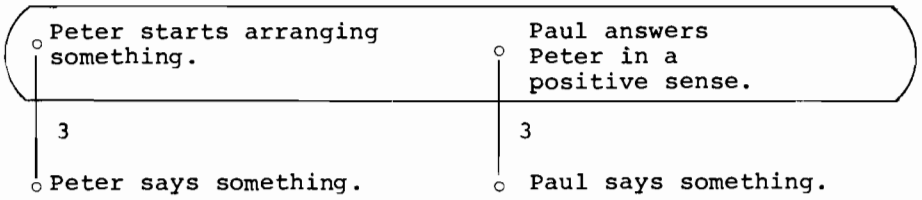
Schank's diagram



Goldman's diagram



Peter and Paul establish an arrangement.



Continuation of the story

DANN BETRITT MARIA DEN RAUM. ALS SIE PAUL SPRECHEN SIEHT, FRAGT SIE, WARUM ER SPRECHE. PAUL ANTWORTET:

- GEMÄSS EINER ABMACHUNG.
- AUF EINE ABMACHUNG HIN.
- ZU EINER ABMACHUNG.
- ZWECKS EINER ABMACHUNG.

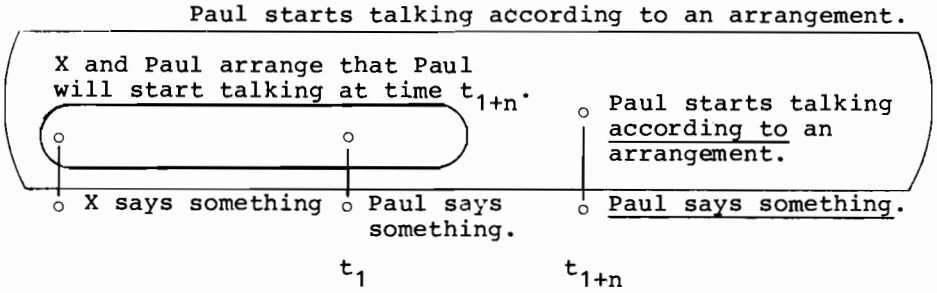
What would be a correct result of understanding the different answers? Note that in the NPs nothing has been said about the agents, the time, and the contents of the arrangement (ABMACHUNG). In the diagrams you will find much more information about the arrangement than is expressed in the NPs. The information can only be found if the features of the subcategorization are used as additional hints during the process of understanding.

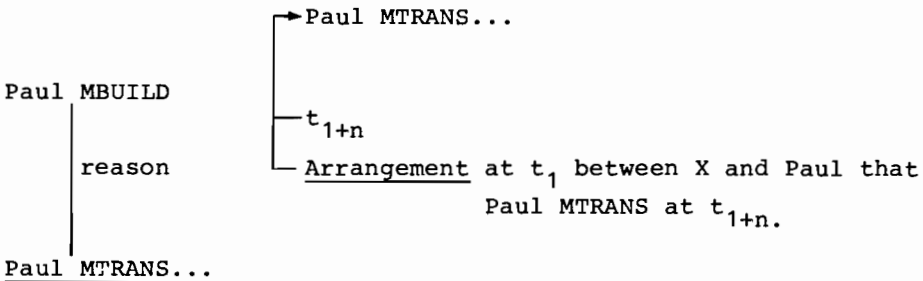
The skeleton of the state/event expressed in the NP is the following:

Agent X and agent Y arrange at time t_1 that agent X/Y will perform an action at time t_{1+n}. ($n \geq 1$)

GEMÄSS (Cf. Table 1.)

Hints for the interpretation: Look for some convention, arrangement, law, etc. in which Paul is one of the agents. The time of the second state/event is before the time of the action being the second state/event. The contents of the convention, etc. must be the same as the action that is the first state/event.

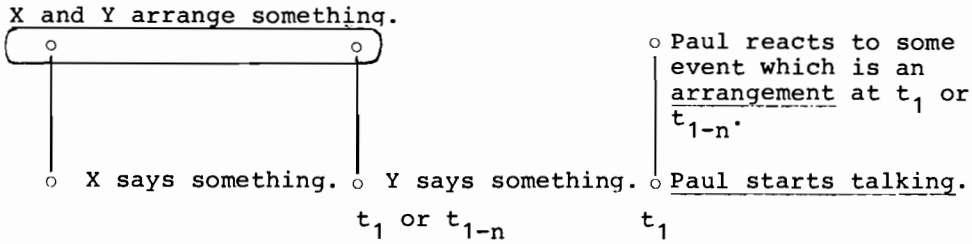


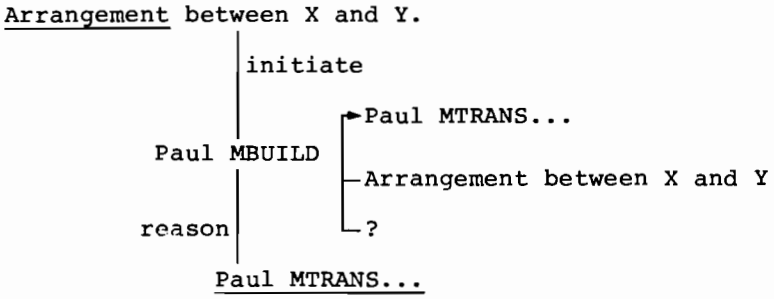


In both diagrams the information verbally expressed is underlined. All other information comes from the specific features of the German preposition GEMÄSS.

AUF ... HIN (Cf. Table 1.)

Hints for the interpretation: Look for the second state/event in the past tense or the present. There need not be any identity of the agents of the first and second state/event.



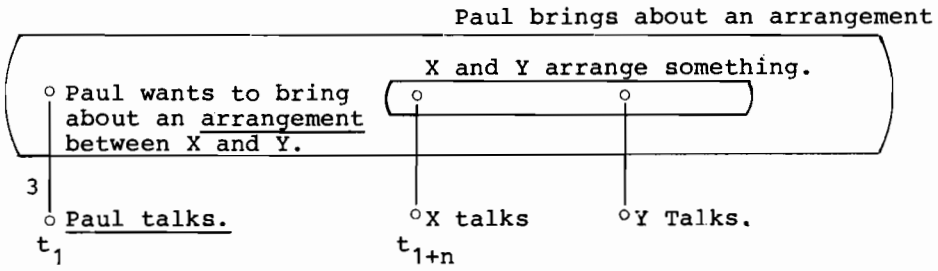


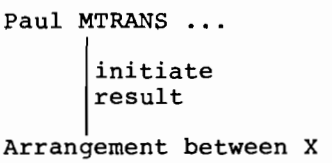
ZU (Cf. Table 1.)

Hints for the interpretation: The first state/event is some conduct of Paul that may be interpreted as an action (= second state/event) of Paul.

ZWECKS (Cf. Table 1.)

Hints for the interpretation: Connect the first state/event (= action) with the intention of the agent to bring about the second state/event. Establish some position of expectance for the second state/event at t_{1+n}.





In our example it was Maria's task to decide with respect to the following specifications of the situation: Who are the agents of the arrangement? When did/does/will the arrangement happen? Is there a logical connection between the contents of the arrangement and the first state/event? Can the arrangement be considered a fact? The quite different structure of the diagrams shows us the differences brought about by the use of different prepositions. All information not underlined could be inferred by means of the features of the subcategorizations of prepositions.

REFERENCES

[1] Piaget, J. *Urteil und Denkprozess des Kindes*, Pädag. Vlg. Schwann, Düsseldorf, 1972.

[2] Maas, U., *Kann man Sprache lehren?*, Rogner & Bernhard, Frankfurt, 1976.

[3] Bobrow, D.G., and A.M. Collins, eds., *Representation and Understanding*, Academic Press, New York, 1975.

[4] Schank, R.C., The Structure of Episodes in Memory, in D.G. Bobrow and A.M. Collins, eds., *Representation and Understanding*, Academic Press, New York, 1975.

[5] Rumelhart, D.E., Notes on a Schema for Stories, in D.G. Bobrow and A.M. Collins, eds., *Representation and Understanding*, Academic Press, New York, 1975.

[6] von Wright, G.H., *Explanation and Understanding*, Cornell University Press, Ithaca, New York, 1971.

[7] Goldman, A.I., *A Theory of Human Action*, Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

[8] Morphosyntaktische Voraussetzungen für eine maschinelle Sprachanalyse des Deutschen, in *Forschungsberichte des Instituts für deutsche Sprache*, Band 18, Ids, Mannheim, 1974.

[9] Schank, R.C., *Causality and Reasoning*, Technical Report 1, Mimeo, Institute for the Study of Semantics and Cognition, Castagnola, Switzerland, 1973.

SYSTEM ASPECTS AND CONSIDERATIONS

A Domain Oriented Natural Language Understanding (DONAU) System for Man-Machine Interaction with Dynamic Data Bases

M. Bernorio, M. Bertoni, A. Dabbene, and M. Somalvico

1. INTRODUCTION

Natural language understanding (NLU) has been for many years one of the central research goals of artificial intelligence (AI) [1,4]. This paper is intended to provide a conceptual and experimental contribution to this field, based on the invention of a particular modular architecture for an NLU system, and centered on the realization and experimentation of DONAU (Domain oriented natural language understanding) system, in a version intended to provide NL programming of a robot [5,13]. The study of the problem of NLU with the goal of obtaining a highly modular system of general validity [6,7,10] is intended to be able to extract from an NL input sentence (IS), information useful for constructing an order to be executed by a robot or a query for a data base [16,17,18].

A preliminary version of the DONAU system has been developed and experimented with on the UNIVAC 1108 of the Milan Polytechnic AI Project (MP-AI Project) [9,12]. At the present time, a DONAU version devoted to the semantic domain (SD) of robotics is working; a new version devoted to the SD of the IIASA data base will be developed next. The DONAU system is based on general criteria, on which there have already been experiments, and which allow the development of different DONAU versions devoted to different SDs.

The DONAU version on robotics has not reached a final stage of development. Study and evolutional changes are being carried out in order to increase and improve the modularization of the system, and to make more practicable and simple any change of SD of application.

The reason the first DONAU version has been devoted to robotics is related to the existence, within the MP-AI Project's Robotics Laboratory, of the SUPERSIGMA robot devoted to the assembly of complex mechanical systems. In the near future, it will be possible to interact with SUPERSIGMA in Italian. The possibility of programming a robot in NL represents an example of an interesting connection between the researches of AI and the technical exigencies of robotics. The SD of robotics in relation to NLU presents a practical and realistic environment (namely industrial robotics) that helps clarify some controversial points about how to face such research problems.

After having chosen the SD of robotics, a simple assembly robot has been simulated on the UNIVAC 1108 computer for the development of a first DONAU version. This simulated robot corresponds in many ways to the SUPERSIGMA robot, so that, in the future, a direct connection between DONAU and SUPERSIGMA should be easily established. The simulator has been developed by using both LISP and MICROPLANNER languages [2,3,14].

The commands that can be executed by the robot are expressed by means of patterns of MICROPLANNER theorems and so this is the control language at present [8,11,15]. It will be possible to replace this simulator with a module of command of SUPERSIGMA later.

In order to develop a new DONAU version to work on a data base, the robot-oriented module will be replaced by another module that can deal with the queries of a data base. After the construction of the robot's simulator, we selected a definitive set of sentences, related to man-machine interaction, called the interaction protocol (IP). The various modules for performing semantic analysis have been constructed with the guidance of the IP. Such goal oriented construction of the semantic analyzer constitutes one of the innovative characteristics of our system. Moreover examination of each IS of the IP enables a deeper understanding of the NL subset related to the actual SD considered by the DONAU system. All the peculiarities, ambiguities, and problems related to NLU are considered in a more precise and realistic way by using IP.

The understanding analysis of an IS is done by syntactic analysis, semantic analysis, operative information extraction, legality control, and finally interaction execution. The syntactic analysis makes use of the PIAF system, developed at the University of Grenoble, France, that works also for the Italian language. To an IS the PIAF system produces as output a few syntactic structures (SSs). The semantic analysis is performed by a semantic discrimination network (DN) based on a hierarchy of model lists (MLs); the DN is an important part of the modular architecture of the DONAU system. A uniform technique has been developed that enables the construction of a different DN for a different SD and, hence, a different IP (i.e. a different DONAU version) is selected. The operative information extraction is performed by another module which operates on the results of the matching between the different SSs and the DN. The results, contained in information lists (ILs), are processed automatically within another module by means of elimination rules based on domain oriented information sets specifying the partial ordering of information content assigned to various words of the lexicon.

Information sets, too, can be changed within a general technique when a new SD, and hence a new DONAU version, is developed. Legality control is made to check that the extracted operative information (OI) is consistent with the interaction world. It

can eliminate operational ambiguities and produces at its output the executable interaction (EI). The module for such control is general; its output is provided to the robot's simulator, which then executes the commands.

In Section 2, we illustrate the role of syntactic and semantic analysis in NLU in a restricted SD. In Section 3, the functional and internal architecture of the DONAU system is presented. In Section 4, a detailed example of understanding of an IS is presented. In Section 5, the experimental results and implementation characteristics of the DONAU system are discussed. In Section 6, the design criteria of a DONAU version for query of data bases are illustrated. Appendix 1, MICROPLANNER characteristics [3], concludes the paper.

2. NATURAL LANGUAGE UNDERSTANDING IN A RESTRICTED SEMANTIC DOMAIN

In this section we describe, in a general way, how structured NLU and the analysis of an NL IS is performed by the DONAU system. The NLU process can be divided into the following phases:

- syntactic analysis;
- semantic analysis;
- operative information extraction;
- legality control; and
- interaction execution.

The syntactic analysis is made by the PIAF system developed at the University of Grenoble [7,8] which works currently for French but can be adapted to Italian, as well. Thus we have organized the syntactic and semantic analyses as two independent activities performed by two autonomous modules [cf. 4]. We have not considered a solution by a close interaction between two analyses, as suggested by Winograd and other authors [1,2], and there is a consequent decrease of efficiency arising from the lack of cooperation between the two analyses. Our decision was influenced by our having already available a module for performing the syntactic analysis; we thus avoided the need to develop such a module ourselves. A more important reason is that the PIAF system can provide for one IS, only a very limited number of output SSs, and so it is not cumbersome to examine all of them. Such a design criterion enables one to proceed with a system that has a modular structure that can be modified and improved without a change on one module influencing another.

The idea of considering the DONAU system as a software system, with all the most advantageous technological results of software engineering research, such as modularization, is one of the main

characteristics of our research. One inconvenience of having syntactic analysis independent of semantic analysis could be that an IS, not permissible at the syntactic level, could be eliminated during the syntactic analysis and it might contain the information necessary for providing a command to the robot. This would be unacceptable, because it would make the formulation of ISs too difficult.

The PIAF system allows nonpermissible ISs to be analyzed within limits and construction of the corresponding SSs. Semantic analysis of such SSs then occurs to effect the understanding process. This PIAF characteristic is a great advantage that has not yet been fully exploited. It will provide in the future interesting improvements.

The complexity of the Italian language makes it impossible for machine understanding at a completely general level. NLU should always be founded on a well defined (and, hence, necessarily formalized) NL subset and on a precise semantic domain (SD) of application. For the first version of our DONAU system we selected the SD of robotics because we have the SUPERSIGMA robot devoted to the assembly of complex mechanical systems in our laboratory. Thus we get a direct NL interaction between the user and the programming of SUPERSIGMA for processing and executing an assembly algorithm. The mechanisms will be discussed in the next section.

The general criterion we used was the adoption of a nondeterministic language for processing the various SSs in order to discriminate which of them should be considered as the correct one, and hence eliminating syntactic ambiguities. We selected MICROPLANNER as a language in which the semantic DN devoted to such disambiguating tasks has been constructed; hence the DONAU system is programmed both in LISP and MICROPLANNER. We have designed the DN under the guidance of a very simple subset of understandable NL, called interaction protocol (IP), made up of a number of model lists (MLs), hierarchically arranged, whose selection and definition is based and guided by the IP. The latter has been defined with the idea of using some typical ISs from which an unambiguous meaning should be obtained and an executable operative information extracted. Obviously, the IP selection should not limit the extension of the Italian subset that can be utilized in the NL interaction. This is true in our case, since the IP is simply a subset of the understandable NL.

Thus our DONAU system is modular and each module is designed with some general technique and with some SD orientation. The IP plays the role of orienting the construction of the DN towards the selected SD. Information extraction is the next phase of the understanding process. It is executed by a separate module whose input is the information constructed by the matching between the various SSs of an IS, and the DN. Such constructed information, contained in ILs is processed to eliminate semantic ambiguities, by this module, which is based on the operation of particular

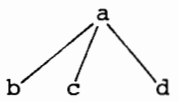
elimination rules. The module is based, like the preceeding two modules devoted to the syntax and semantic analyses, on an orientation towards the selected SD. Such guidance is provided by information sets dependent on the SD, that specify the application of the elimination rules. We obtain as the output of this module operative information (OI) which corresponds to interactions with the real world (in our case commands to a robot, but, in another case, queries to a data base).

The next phase, called the legality control, is performed by another module which processes the OI and eliminates the operative illegalities in those ISs that are correct both at the syntactic and the semantic level, but that are inconsistent with the actual status and configuration of the interaction world (the robot or the data base). This module, too, is strongly related to the selected SD. The output, EI, is processed by the last module of the DONAU system, which, in our DONAU version on robotics, is the robot simulator, but later will be the interaction execution module connected to the SUPERSIGMA robot.

The syntactic analysis is performed by the PIAF system, while the successive phases (semantic analysis, etc.) are carried on by the DONAU system. While the DONAU activity will be discussed in greater detail in the following section, it is worthwhile to illustrate here the output of the PIAF system, which constitutes the input of the DONAU system at present.

The PIAF system receives the IS to be analyzed, and provides as its output a small number of SSs as alternative candidates of the syntactic interpretation of the IS.

It is the task of further DONAU semantic analysis to choose the one correct candidate, not only syntactically, as anyone of the SSs, but semantically as well, as the selected SS is. While an SS, actually outputted by the PIAF system, is a tree, the input to the DONAU system is a parenthetical description of the same tree. Thus to the tree:



will correspond the list (a b c * d), i.e. a list defined recursively such that the first element is the root of the tree, and the asterix separates the left subtrees from the right ones. The PIAF system constructs an SS in such a way that the root of a tree corresponds to an NL word which, in the IS, is preceeded by words contained in the left subtrees and is followed by the words contained in the right subtrees.

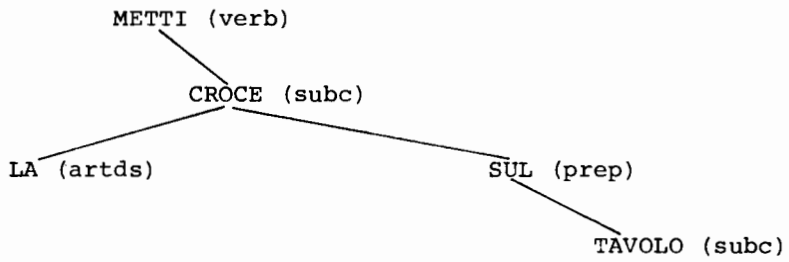
In order to illustrate the output of the PIAF system in a better way, let us examine the following simple example of IS:

METTI LA CROCE SUL TAVOLO

(which, in Italian, means: put the cross on the table).

The PIAF system provides, as its output, the following two SSs:

((METTI verb)*((CROCE subc) (LA artds)*)((SUL prep)*(TAVOLO subc)))



((METTI verb)*((CROCE subc)(LA artds)*((SUL prep)*(TAVOLO subc))))

In this example of IS, we can identify three parts: the verb (METTI (put)), the object (LA CROCE (the cross)), and the place (SUL TAVOLO (on the table)). It is easy to see that only the first of the two SSs, presents a clear identification of these three parts.

These are the criteria that will guide the algorithms devoted to the semantic analysis, as will be illustrated in the next sections.

3. ARCHITECTURE OF THE DONAU SYSTEM

In this section we describe the architecture of the system from two points of view: one related to the functional aspects of the system, and the other to the implementation and program organization of the system. As we have seen we distinguish between syntactic and semantic analysis. The PIAF system does the first analysis and provides, at its outputs, some SSs in the form of LISP lists. Whenever the IS has a syntactic ambiguity, PIAF

gives as output more than one SS. The semantic analysis does the necessary discrimination between the alternative SSs.

To achieve this, the semantic analysis has been programmed within a nondeterministic programming language MICROPLANNER.

Figure 1 shows the logical organization on which the DONAU system is based. The input of the DONAU system is the set of the SSs output from the PIAF system. The output of the DONAU system is the assembly algorithm for the robot and the answers to the questions; in the case of misunderstanding of the input, some diagnostic message can be given as output as well.

In the MICROPLANNER problem base we find the morphological model (MM) given by the PIAF system and inserted with the PIAF/MICROPLANNER translator. The semantic model (SM) is divided in two blocks: the functional model (FM) and the interaction model (IM).

The FM is the fundamental part of the SM and is permanently inserted within the system and represents the robot's world and the SD in which we operate. If a different SD, like the management of a data base is chosen, the FM will contain the structure and the model of the data base. At present, the FM is inserted directly in MICROPLANNER, but, in the future, it will be inserted in NL through PIAF and a PIAF/MICROPLANNER translator. In the case of robotics, the FM contains that fundamental information which concerns the part of the world the robot cannot change. The FM also contains the information necessary to solve syntactic ambiguities, i.e., to select one SS within the input set of SSs corresponding to a given IS, and includes the knowledge adopted for extracting, from the chosen SS, all the OI useful to communicate in a formal language a command to be executed by the robot.

The IM contains all the information on the fundamental mechanical component parts (called modules) used by the robot during the assembly process. The content of the IM is continually added to by inserting descriptions of new component parts that are built by the robot within the man-machine interaction. The IM includes information useful to answer questions asked of the system. The MICROPLANNER interpreter makes use of the information in the problem base and does semantic analysis to produce OI providing a command to the robot. It, too, modifies and enriches the IM. It answers questions and these are translated into NL by a MICROPLANNER/NL translator.

Thus, from an information processing standpoint, the SM is a unique model; a strong interaction exists between the FM and the IM.

Figure 2 shows the program architecture of the DONAU system. The modular organization and the division into different blocks is clearly shown. The input is the set of SSs corresponding to the IS, which are discriminated by the filter block (1) written in MICROPLANNER. The filter eliminates syntactic ambiguities and

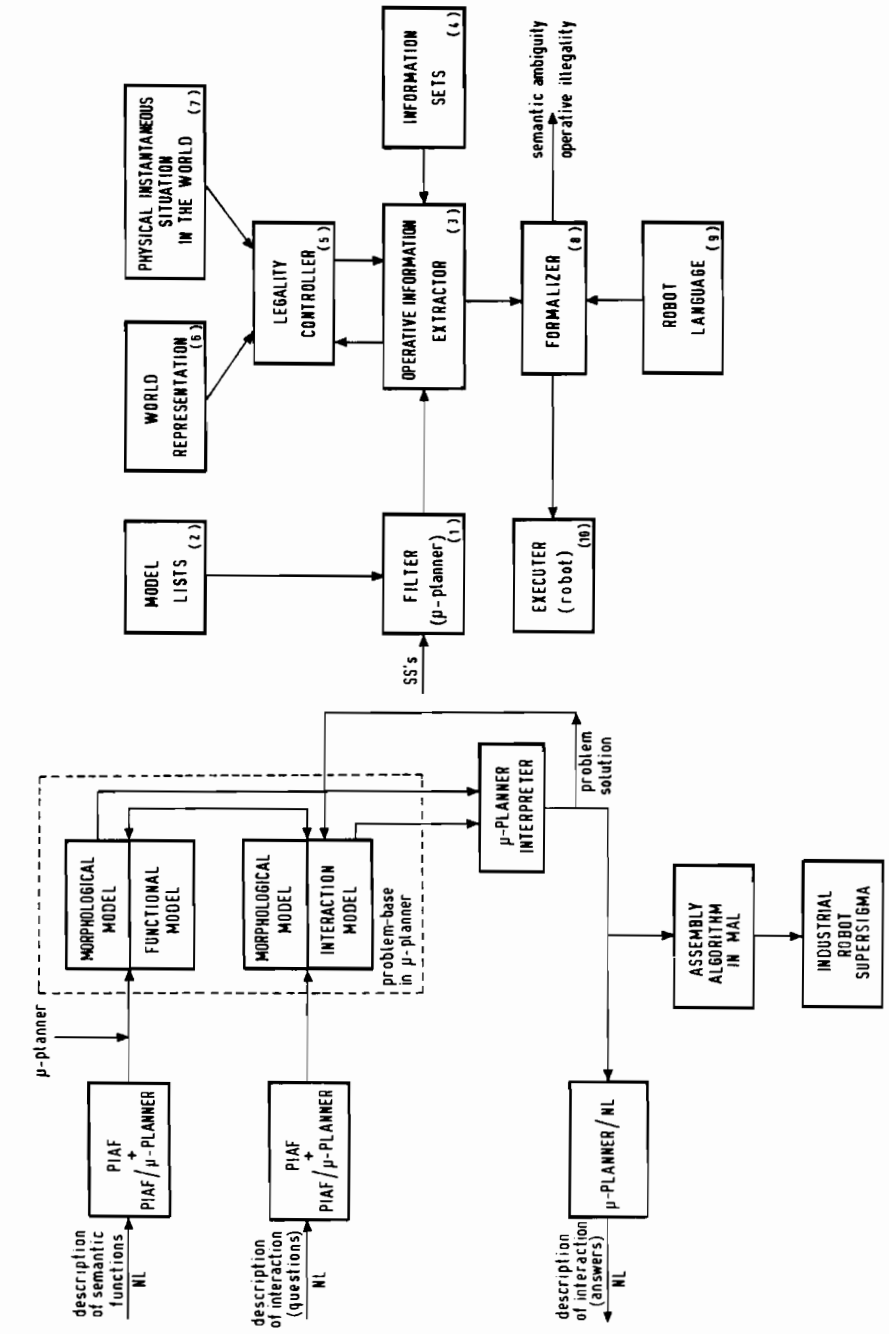


Figure 1. Logical organization of the DONAU System.

Figure 2. Program architecture of the DONAU System.

selects one SS by using block (2), which contains the MLs and which is the DN constructed under the guidance of the IP. From the matching between the discriminated SS and the DN, the ILs are constructed and are provided as input to block (3), the operative information extractor. This eliminates the semantic ambiguities and extracts from the ILs the OI by using block (4), which contains the ISs constructed under the guidance of the IP. The result obtained from the extraction activity is provided as input to block (5), the legality controller. This block verifies the consistency between the OI and the SD oriented contents of block (6), called world representation, and of block (7), called physical instantaneous situation in the world, which contain both the static and the dynamic descriptions of the robot's functional ability and operative activity. When this control succeeds, the EI is obtained and is provided as input to block (8), called formalizer, which, by means of block (9) containing the robot language description, constructs the executable program for block (10), called executer, which corresponds to the robot. When the processing activities of either block (3) or block (5) fail, an NL diagnostic message, related either to a semantic ambiguity or to an operative illegality, is provided by the block (8) as well.

It is important to note that the program architecture of the DONAU system is highly modular. Blocks (2), (4), (6), (7), (9), and (10), related to a specific SD, have to be changed when a new SD is selected. However, either the new SD or the corresponding new selected IP provide guidance for constructing, within the use of the same general techniques, these SD oriented blocks. Blocks (1), (3), (5), and (8), related to different phases of the NL understanding process, are of general validity and do not have to be changed when a different SD is chosen.

At present, we are working on enhancing the modular program architecture of the DONAU system, either by improving the definition of each module or by clearly specifying the interfaces between every two connected modules. Thus advantage can be taken of all technological improvements proposed by software engineering.

4. THE ANALYSIS OF NL INPUT SENTENCES

In this section we discuss how the problem of the extraction of the OI from an IS can be solved.

As the DONAU's input we have some SSs, provided by PIAF, that are represented for our convenience as LISP lists. We have to elaborate these lists in such a way that we can recover the useful elements for the construction of a MICROPLANNER command for the robot. Before examining in detail the mechanism that guides the semantic analysis, we want to point out how in an IS that is apparently clear and easily understandable for the human intelligence there can quite often be ambiguities and lack of

information. This is because man often expresses himself with either sentences containing redundant information or with sentences requiring some deductive activity. The computer, on the other hand, in order to operate in a correct way, needs precisely described information--neither inadequate nor redundant. It is therefore necessary to insert in the modules that deal with the analysis of the SSs, complete knowledge of the world in which the robot operates. This will allow the computer to understand the IS, i.e., to make the necessary deductions from the information contained in the IS.

We will now show how the analysis of the SSs has been realized. First, we discriminate from the set of SSs provided by PIAF a certain number of parenthetical structures; second, we extract from such structures sufficient information from the IS to build the command to the robot; third, we make the extracted information consistent with the physical reality of the robot's world.

Let us examine a simple example of this process. The following IS to be analyzed:

PRENDI UNA TI DAL TAVOLO
(take a TI from the table)

The first word the filter will consider is the verb; in our case: PRENDI (take). Associated with this verb are some MLs corresponding to possible structures of a command beginning with this verb; here, the following two MLs:

(OBJ)
(OBJ PLACE)

The MLs indicate, as elementary semantic models, that the verb PRENDI must necessarily be associated with the information of the object to be taken and optionally with the information of the place where the object to be taken is located.

In our example, we have two groups of words corresponding to OBJ and to PLACE. One SS provided by PIAF is:

(PRENDI*(TI UNA*) (DAL*TAVOLO))
(take *(TI a *) (from-the*table))

The elements OBJ and PLACE, belonging to the considered ML, are called meaningful elements (MEs), each one of which is associated with a set of MLs which indicate the various ways in which they can be further developed. We have the following correspondences:

OBJ → (SUBS ART)
PLACE → (PREP SUBS)

The elements SUBS, ART, and PREP cannot be further developed; therefore they are called basic meaningful elements (BMEs).

At this point, it is clear how we can make a first choice of the SSs given by PIAF. Each SS is examined, and at each level of the syntactic tree, it is matched with the MLs equal in length to the number of syntactic subtrees whose roots belong to that level. This matching procedure is repeated recursively at all levels of the SS, and, if the matching succeeds, then the SS examined is the correct one.

The set of MLs has been defined on the basis of the example IP, but is has a more general validity, because it enables the understanding of a large set of ISs which includes the IP as a small subset of its own. Neither the insertion of new MLs, nor the modification of already existing ones presents any excessive difficulty.

We will now discuss how the filter's discrimination process produces the ILs that constitute the basis for the continuing extraction of the OI. During the semantic analysis, each BME encountered is used for further processing. Indeed, each BME obtained guarantees the IS is so far correct. It contains information very useful for building a command to the robot. The question now arises of identifying the BMEs that are really of interest, and of how to process their useful information. To solve these problems, we associate to each BME encountered a property that depends on:

- the position of the BME, namely the ME from which the BME has been derived;

- the group of Italian language words to which the BME belongs (e.g., proper noun, common noun, couple of coordinates, etc.).

Particular lists, namely the ILs, are associated with these and each BME is inserted in the corresponding IL.

Intuitively, we identify two categories of BMEs: those that provide information related to the considered part of the SS (such BMEs--e.g., prepositions--give some information on which MLs have to be examined in the sequence of the discrimination process to decide whether the considered SS is a correct one); and those that contain the information necessary to formalize a command to the robot. Only the latter are inserted in the ILs.

We can examine a simple example in order to illustrate the content of the ILS. The following IS is to be analyzed:

PRENDI IL MODULO TI DAL TAVOLO
(take the module TI from-the table)

The correct SS, given by the PIAF system, is the following:

(PRENDI*(MODULO IL*TI) (DAL*TAVOLO))
(take * (module the*TI) (from-the * table))

The correctness of this sentence is controlled by using the MLs as previously illustrated. The semantic analysis will eventually provide a certain number of BMEs that we briefly discuss. MODULO (module) and TI (TI) are BMEs which both satisfy the same property of deriving from the same ME OBJ, and of belonging to the group of substantives in the lexicon of the system. Because of this latter property, P_1 the two BMEs are inserted in a corresponding IL, IL_1, which contains all the information concerning the object of the action. IL (the) is a BME that satisfies a different property, P_2, since it derives from the same ME OBJ, but belongs to the group of articles in the lexicon of the system; therefore, it is inserted in a different IL, IL_2. The BME TAVOLO (table) again satisfies the property of belonging to the group of substantives, like MODULO (module) and TI (TI). However, it has been derived from the ME PLACE, and, therefore, it is inserted in a new IL, IL_3, which contains all the information on the place of the action. The BME DAL (from-the) is a preposition and it is utilized only in order to more easily identify the MLs associated with the considered SS; it is inserted in a new IL, IL_4. Thus, when we have completed the examination of the correctness of the SS, we have constructed the following ILs:

IL_1 = (MODULO TI) (module TI)
IL_2 = (IL) (the)
IL_3 = (TAVOLO) (table)
IL_4 = (DAL) (from-the)

IL_1 and IL_2 are of particular interest for the construction of the command to the robot. Even in this simple example the information obtained is still in a rather complex form, mainly because two BMEs belong to the same IL_1.

In order to construct the command, it is necessary to isolate one object of the action performed by the robot. Therefore, the system should be able to recognize the fact that TI is a particular module and, hence, that it contains all the information

required for providing the command to the robot. This activity, executed by the operative information extractor, is performed by utilizing a set of ISs; each IS is associated with one BME, a substantive, and it isolates the information associated with that BME.

In our example, the two BMEs TI (TI) and MODULO (module) are associated with the two following ISs:

TI → {TI, MODULO,PEZZO} {TI,module, piece}

i.e., each TI is a module and is a piece;

MODULO → {MODULO,PEZZO} {module,piece}

i.e., each module is a piece. By simple set inclusion operations on the ISs, the OI can be extracted on the basis of the ILs. More precisely, a binary relation of dominance $<$, defined on the set of BMEs, is introduced. Given two BMEs, x_1 and x_2, we say that $x_1 < x_2$ when the two corresponding ISs, IS_1 and IS_2, satisfy the relation $IS_1 \subset IS_2$. Hence x_1 is dominated by x_2, i.e., x_2 has a greater information content than x_1. This dominance relation is iteratively applied to the BME of the same IL, by eliminating each time a BME dominated by another one. Thus, in our example, since MODULO (module) is dominated by TI (TI), we obtain the following new IL_1:

IL_1 = (TI) (TI)

In order to verify the correctness of the OI extracted from the IS, and to construct the order for the robot, the system must know the world in which the robot operates and its situation at that moment. Thus the legality controller, which checks the correctness of the OI, must be strictly dependent on the particular SD chosen.

To decide whether the OI is correct, we start the processing activity again by considering the verb contained in the IS. Each verb is associated with some MLs, which characterize the order to the robot related to the use of that verb. In our example, PRENDI (take) is related to the following ML:

(OBJ PLACE)

We try to construct this type of order with the use of the OI. If it can be done, the OI is sufficient if not, more detailed information will be requested.

Controlling the sufficiency of the OI is not necessarily the only activity of the legality controller: the consistency of the OI with the situation of the world at that time has to be checked. For instance, the following IS:

PRENDI LA TI DAL TAVOLO
(take the TI from-the table)

leads to the construction of the following OI:

(PRENDI TI TAVOLO)
(take TI table)

Because of the article LA (the), we also have to verify that there is only one TI (TI) on the table so that the order can actually be executed. Then the OI is legal and it becomes the EI ready for execution.

Whenever the OI is controlled and found to be sufficient and consistent, it becomes the EI communicated to the formalizer which, by using robot language, provides the command to the executor, i.e. the robot, for execution.

In conclusion, let us consider what happens when more than one SS is accepted by the filter:

- Either all the accepted SSs lead to the construction of the same command;
- Or different SSs correspond to different commands.

In the first case, we have ambiguity only at the syntactic level; it is disambiguated by the next semantic analysis. In the second case, the ambiguity is also at the semantic level, i.e. the IS actually has more than one meaning; here the system asks for more information by providing an error message of semantic ambiguity.

In a similar way, when the OI examined by the legality controller is either not sufficient or not consistent, an error message of operative illegality is provided.

5. EXPERIMENTAL RESULTS

The experimental activity developed has first been devoted to the simulation of the robot (that is, to the construction of the block 10, the executor, of Figure 2). However, all the

various blocks of the DONAU architecture, shown in Figure 2 have been implemented successfully. Here we describe the characteristics of the executor; an example of interaction is provided in Appendix 1.

The task of the simulated robot is the assembly process of a mechanical system (and, if necessary, of complex component pieces); the assembly takes place on the table (TAVOLO). The components that can be handled by the robot are shaped with elements oriented horizontally or vertically with respect to the table. Thus, the table is a plane divided into an array; each segment of the array has the length of the element (LATO) by which a piece (PEZZO) is made up. Each element of the array can be uniquely defined by two Cartesian coordinates (x,y). The dimensions of the array are usually assumed by the executor to be of fixed value; it is however possible to modify such dimensions during the robot's activity.

The placement or the replacement of a piece on the table is considered as being composed of two distinct actions, namely the grasping of the piece by the robot's hand (MANO), and the subsequent repositioning of the piece on the table.

A stand-by place (RIPOSTIGLIO) is also available to the robot for the temporary placing of pieces (or of subpieces during the assembly of a piece), and in certain cases the robot can place a piece in its hand in the stand-by place without being explicitly ordered to do so.

Usually, the assembly process of a mechanical system is executed by using particular pieces, typically of simple structure, called modules, available to the robot in fixed places called boxes. As each type of module (MODULO) is arranged in a corresponding box (SCATOLA), only the type name has to be indicated to the robot. The robot can only grasp a module from box; it cannot put it back. Thus, the box simulates the loader of one component part of a mechanical system, which will be utilized during the assembly process.

The hand enables the robot to grasp a piece, to rotate it 90°, 180°, or 270°, to position it on the table or in the stand-by place, and to connect (by screwing or welding) and disconnect (by unscrewing).

In order to connect or disconnect, it is sufficient to indicate only the place (i.e., the two coordinates), where such operation has to be executed. The executor automatically controls the legality of the operation before executing it. The hand can grasp only one element of a piece and, if no other specific indication is provided, the piece is grasped at the first element mentioned in the piece's internal description. The hand can also eliminate pieces by putting them in the waste place, where they cannot be grasped anymore.

The internal description of a piece (or of a module) is done in such a way that the piece is thought of as occupying some segments of the table, indicated by increasing integers. One element of the piece is selected as corresponding to the first integer in the internal representation, and is the standard reference element for the operation of placement on the table. In fact, the position (i.e. the two coordinates) indicated in the placement of a piece corresponds to the position of the standard reference element of that piece, which corresponds to the beginning of a path that completely covers the piece. The increasing integers reproduce the order in which the path covers the piece.

The path is described with a list made up of the operators L (left), R (right), U (up), and D (down). When a fork is encountered, sublists are opened within a given list. Thus the module TI, which has the shape indicated by its name can be considered as made up of three elements connected by the path (R (D) (R)). The first operator (R) corresponds to the standard reference element of the module TI (i.e., the leftmost element). Since the user does not necessarily know what the internal representation of a given piece is, it is possible for him to query the robot in NL for having such an internal representation as answer. This answer, called the characteristics (CARATTERISTICHE) of the piece (or of the module), is made up of the correspondence between the segments of the path covering the piece, and the increasing integers.

Proper names can be attributed to pieces (and to modules) for better identification and handling. Similarly a particular element of a piece (or of a module) can be given a proper name (corresponding to a particular integer of the characteristics of the piece).

The description of a module is inserted in the MICROPLANNER problem base (see Figure 1) by means of an assertion. For instance, in the case of the module TI:

```
(THASSERT (TI (R(D)(R))))
```

Also the descriptions of the pieces' positions on the table and of the executed assembly operations are entered in the MICROPLANNER problem base by using assertions. When a piece is already on the table, all the information concerning the piece is bound to the particular position which the piece occupies. Thus, the processing of assembly operations related to the piece is made in a very efficient way. But on the other hand, this fact can make a change in this information difficult, when the piece's position changes during the assembly process.

To solve this problem, the information on the execution of an assembly operation is structured in a way that makes it independent from the piece's positions. The information on a piece

on the table is transferred in specifically designed transport lists when the piece is grasped by the robot's hand. When the piece is repositioned by the hand in a new place on the table, the piece's information, stored in the transport lists, is modified appropriately. The request for the execution of a command to the robot corresponds to the associative call of a corresponding theorem contained in the MICROPLANNER problem base.

We will now illustrate briefly an example of interaction. We will first give here the sequence of ISs that constitute the example of interaction:

PRENDI LA TI DALLA SCATOLA
(take the TI from-the box)

METTI LA TI NEL POSTO (9 4)
(put the TI in-the place (9 4)

PRENDI LA TI DALLA SCATOLA
(take the TI from-the box)

METTI LA TI SUL TAVOLO
(put the TI on-the table)

PRENDI LA TI DAL TAVOLO
(take the TI from-the table)

PRENDI UNA TI DAL TAVOLO
(take a TI from-the table)

PRENDI IL MODULO TII CHE E' NELLA SCATOLA
(take the module TII which is in-the box)

DAMMI LE CARATTERISTICHE DELLA TII
(give-me the characteristics of-the TII)

CHIAMA IL LATO 2 DELLA TII CHE E' in MANO PIPPO
(call the element 2 of the TII which is in hand PIPPO)

AVVITA LA TI E LA TII NEL POSTO ((1 2))
(screw the TI and the TII in-the place ((1 2)))

SALDA IN ((3 2))
(weld in ((3 2)))

CHIAMA CROCE IL PEZZO CHE E' SUL TAVOLO
(call CROCE the piece which is on-the table)

The first command is grasping the module TI from the box. Then it is placed with its standard reference element in position (9 4) of the table, and a second module TI is grasped from the box. This time, the command of positioning it on the table

does not indicate a specific position, so the robot automatically places the module in the lowest leftmost free position of the table--position (1 2) of the table. The following command of grasping the (LA) TI from the table, is not executed by the robot, which signals the existence of an operative ambiguity; since two TI's are already placed on the table the article the is ambiguous. However, the next command, of grasping a (UNA) TI, can be executed: the robot automatically grasps the TI placed in (9 4). Then the command of grasping the module TII (i.e., a module, similar to TI, but with a right element of double length) from the box is given. Since the hand is already holding the TI, the robot first eliminates this TI, by automatically putting it in the waste place, and then grasps the TII from the box. Note that the legality controller will check that TII is indeed a module, as stated in the IS. Next, the robot provides, upon request, the characteristics of the TII so that the user can make reference to a particular element of this module. The following command is naming element 2 of the TII PIPPO. Then the TII is positioned on the table with PIPPO in place (3 2). This is a new way of specifying where to position a module, because the indicated place is not one of the standard reference elements of the module. The next two orders are for connecting TI and the TII, first by screwing in place (1 2), and second by welding in place (3 2). At this point, the two previously distinct modules are now connected together irreversibly, thus yielding a new piece which, with the last command, is named CROCE. All the commands encountered in this example of interaction are either physical or non-physical commands, depending on the involvement of actual movements of the hand and pieces, and on the execution of assembly operations.

6. TOWARD A DONAU VERSION ON DATA BASE QUERY SYSTEM

We will now discuss how to build a DONAU version for querying, in Italian, a data base. This problem appears to be very similar to that of programming a robot. The various phases of the NL understanding process are easily applicable to this new and very important use. Moreover, as it has been illustrated in Section 3, the architecture of the DONAU system (see Figure 2) is highly modular. We have clearly divided the SD independent blocks (1), (3), (5), and (8)) from the SD dependent blocks ((2), (4), (6), (9), and (10)). The latter blocks need to be changed and new ones, oriented toward the new SD of data bases, have to be developed and put in their place. In this way, a new DONAU version can be built by utilizing a large part of the old DONAU version. For both robotics and data bases, it is convenient to divide the ISs of an interaction into the following three classes: orders, questions, and descriptions. Orders relate to the specification of the activity that the artificial system (either a robot or a data base) has to perform (either an assembly operation or a data input/output. Questions refer to the request to the artificial system for some characteristics and information about its structure and status needed by the user for a better knowledge of its

artificial behavior and available operating procedure. Descriptions indicate the specifications and modifications that the user operates on the structure and configuration of the artificial system.

As we have illustrated in Section 3, the first step to be accomplished to develop the SD dependent blocks, is the identification of a good IP. It is important to choose a set of typical ISs that are examples of common interaction--in our case queries that the new DONAU version can deal with.

We will now illustrate an example of an IP oriented to data bases. This example is reduced to a minimum for providing an understanding of its main characteristics.

DAMMI LA PRODUZIONE DI PETROLIO DEL KUWAIT DEL 1975
(give-me the production of oil of-the KUWAIT of-the 1975)

DAMMI LA PRODUZIONE DI PETROLIO DEL 1975
(give-me the production of oil of-the 1975)

DIMMI QUALE E' IL MAGGIOR PRODUTTORE DI PETROLIO
(tell-me which is the greatest producer of oil)

DAMMI LA PRODUZIONE DI PETROLIO DEI PAESI ARABI
(give-me the production of oil of-the Arab countries)

DIMMI QUANTI SONO I PRODUTTORI DI PETROLIO
(tell-me how-many are the producers of oil)

Careful study of such an IP, together with the analysis of the SSs provided by PIAF corresponding to each IS of the IP, constitutes the basis for the design of block (2), i.e. for the selection of the MLs that provide the semantic representation of the SD. The MEs that appear in the MLs related to the new SD of data bases are almost the same ones that have been adopted in the MLs related to the old SD of robotics, but the BMEs of these new MLs are almost completely different.

We shall now examine two ISs, and illustrate how the DONAU version on data bases will perform the NL understanding process on these two examples:

DAMMI LA PRODUZIONE DI PETROLIO DEL KUWAIT DEL 1975
(give-me the production of oil of-the KUWAIT of-the 1975)

DAMMI LA PRODUZIONE DI PETROLIO DEL 1975
(give-me the production of oil of-the 1975)

While the first query requests only the oil production of Kuwait in 1975, the second query asks for the oil production of the whole world in 1975. The correct SS given by the PIAF system for the first IS:

```
(DAMMI*(PRODUZIONE LA*(DI*PETROLIO)(DEL*(KUWAIT*(DEL*1975)))))
(give-me*(production the*(of*oil)(of-the*(KUWAIT*(of-the*1975)))))
```

The first ML associated to DAMMI (give-me) is:

```
(OBJ)
```

This ML indicates that the verb DAMMI must be associated with the information to be given. In fact, in the considered SS, the root of the tree (i.e. DAMMI) is followed by only one subtree, namely a right subtree, which corresponds to the only ME OBJ of the matched ML.

Next, the DONAU system selects the following MLs when trying to match the considered SS with the hierarchical arrangement of the MLs:

```
OBJ  → (SUBS ART SPEC PLACE-SPEC)
SUBS → (PRODUZIONE)    (production)
ART  → (LA)        (the)
SPEC → (PREP    SUBS)
PREP → (DI)         (of)
SUBS → (PETROLIO) (oil)
PLACE-SPEC → (PREP    PREP-SPEC)
PREP → (DEL)   (of-the)
PREP-SPEC → (SUBS     TIME-SPEC)
SUBS → (KUWAIT)       (KUWAIT)
TIME-SPEC → (PREP    SUBS)
PREP → (DEL)   (of-the)
SUBS → (1975)         (1975)
```

It is easy to ensure that the topology of the discrimination tree (i.e. the hierarchical organization of the previously indicated MLs) completely matches the topology of the SS (i.e. the corresponding syntactic tree described parenthetically--see Section 2).

The correct SS given by the PIAF system for the second IS is:

```
(DAMMI*PRODUZIONE LA*(DI*PETROLIO) (DEL*1975)))
(give-me*(production the*(of*oil)(of-the*1975)))
```

Again the first ML associated to DAMMI (give-me) is:

(OBJ)

and the DONAU system selects the following MLs which completely match the SS when hierarchically arranged:

```
OBJ  → (SUBS ART SPEC TIME-SPEC)
SUBS → (PRODUZIONE   (production)
ART  → (LA)          (the)
SPEC → (PREP SUBS)
PREP → (DI)          (of)
SUBS → (PETROLIO)    (oil)
TIME-SPEC → (PREP SUBS)
PREP → (DEL)         (of-the)
SUBS → (1975)        (1975)
```

From this semantic analysis, the DONAU system constructs the ILs as described in Section 4. The following ILs correspond with the first IS:

IL_1 = (PRODUZIONE PETROLIO) (production oil)
IL_2 = (KUWAIT) (KUWAIT)
IL_3 = (1975) (1975)

The following ILs correspond with the second IS:

IL_1 = (PRODUZIONE PETROLIO) (production oil)
IL_2 = (NIL)
IL_3 = (1975) (1975)

In IL_1, it is clear that the BME PETROLIO (oil) also contains the information expressed by PRODUZIONE (production). The operative information extractor (see Section 4) by applying the following dominance relation:

PRODUZIONE < PETROLIO
(production < oil)

reduces IL_1 to

IL_1 = (PETROLIO) (oil)

Thus to correspond with these two ISs, the OI and, after the activity of the legality controller, the EI is provided to the formalizer as the two following patterns of MICROPLANNER theorems:

(PETROLIO KUWAIT 1975)
(oil KUWAIT 1975)

(PETROLIO NIL 1975)
(oil NIL 1975)

We have assumed the internal formal query language of the data base is MICROPLANNER. If a different formal language is utilized, then the formalizer will present the EI appropriately.

REFERENCES

[1] Winograd, T., *Procedures as Representation of Knowledge in a Computer Program for Understanding Natural Language*, MAC-TR-84, Project MAC, MIT, Cambridge, Mass., 1971.

[2] Woods, A., Transition Network Grammars for Natural Language Analysis, *Comm. ACM*, 13 (1970), 591-606.

[3] Hewitt, C., *Procedural Embedding of Knowledge in Planner*, Memo, AI Lab., MIT, Cambridge, Mass., September 1971.

[4] Erchov, A., P. Mel' Chuk, and N. Nariniany, *RITA - An Experimental Man-Computer System on a Natural Language Basis*, presented at the 4th IJCAI, Tbilisi, August 1975.

[5] Bullwinkle, C.L., *Picnics, Kittens and Wigs: Using Scenarios for the Sentence Compilation Task*, presented at the 4th IJCAI, Tbilisi, August 1975.

[6] Miller, P.L., *An Adactive and Natural Language System that Listens, Asks, and Learns*, presented at the 4th IJCAI, Tbilisi, August 1975.

[7] Courtin, J., *Un Systeme d'Analyse des Langues Naturelles: Application à la Correction Interactive de Textes*, Université de Grenoble, Laboratoire de Informatique, Grenoble, 1973.

[8] Grandjean, E., *System PIAF Detecteur de Fautes d'Ortographe*, PIAFDET, Université de Grenoble, Grenoble, 1974.

[9] Gini, G., M. Gini, and M. Somalvico, *Emergency Recovery in Intelligent Industrial Robots*, presented at the 5th ISIR, Chicago, September 1975.

[10] Schwind, C., *Generating Hierarchical Semantic Networks from Natural Language Discourse*, presented at the 4th IJCAI, Tbilisi, August 1975.

[11] Yoshida, S. *On the System of Concepts Relations and Outline of the Natural Language Systems*, presented at the 4th IJCAI, Tbilisi, August 1975.

[12] Bernorio, M., M. Bertoni, A. Dabbene, and M. Somalvico, Interazione Uomo Macchina in Linguaggio Quasi Naturale, in *Proceedings Annual AICA Conference*, Milan, October 1976.

[13] Creg, W., *Scragg, Answering Process Questions*, presented at the 4th IJCAI, Tbilisi, August 1975.

[14] Hewitt, C., *How to Use What You Know*, presented at the 4th IJCAI, Tbilisi, August 1975.

[15] Courtin, J., *Organization d'un Dictionaire pour l'Analyse Morphologique*, presented at the Seminaire de Théorie des Automates et Traitment Automatique des Langues, IRMA, Grenoble, 1973.

[16] Hays, D.G., *Dependency Theory: A Formalism and Some Observation*, Memorandum PM-4087-Ph, The Rand Corporation, Santa Monica, California, 1974.

[17] Veillon, G., *Modèles et Algorithmes pour la Traduction Automatique*, Thèse d'Etat, Université de Grenoble, Grenoble, 1970.

[18] Bernorio, M., M. Bertoni, A. Dabbene, and M. Somalvico, Quasi Natural Language Understanding in the Semantic Domain of Robotics, in *Applied Robotics 77*, Plsen, Czechoslovakia, October 1975.

Appendix 1. Illustration of the MICROPLANNER Language

The programming language PLANNER was designed by Carl Hewitt as a goal-oriented procedural language. It operates on a set of assertions and it has a special mechanism for handling them efficiently. It can include every complex information that can be expressed in first order predicate calculus and, more generally, in ω-order logic. It is typical of PLANNER that complex information is represented in the form of procedures, called theorems, that can include all types of knowledge expressed in the best way for proving statements, i.e. for reaching a goal.

Since PLANNER is goal-oriented, it does not have to deal with all the details of different procedures. For example, if we have to verify a statement, it is not necessary to specify which theorem the system must use. This is done automatically by the PLANNER interpreter which operates an associative call, called pattern matching, of theorems and assertions. However, we can indicate which procedure should be used first to get a faster answer.

When a new theorem is added to the problem-base (i.e. the set of stored theorems and assertions), the system can use it without further specification. PLANNER theorems can be written independently of each other, without concern about how or when they will be called, or about what other theorems and assertions will be needed for reaching a goal or subgoal. The great advantage of using PLANNER is the possibility of representing complex information by a procedural problem base.

The MICROPLANNER language is a subset of PLANNER implemented by Winograd, Charniak, and Sussman. The best way to understand MICROPLANNER is to see how it operates on a simple example. Consider the following sentences:

(a) The TI is a module;
(b) All the modules are on the table;
(c) The TI is on the table.

In order to assert sentence (a), we can enter:

```
(THASSERT (MODULE TI))
```

In this MICROPLANNER instruction, THASSERT is a function that simply adds its argument to the problem-base. Sentence (b) is

entered as a theorem in the following way:

```
(PUT THEOREM1 THEOREM (THCONSE (X)
      (TABLE $?X)
   (THGOAL (MODULE $?X))   ))
```

where $?X means that X is a variable. Now our goal is to prove statement (c), i.e.

```
(TABLE TI)
```

We use a THCONSE theorem, i.e. a consequent theorem, which means that when we want to prove a goal of the form:

```
(TABLE $?X)
```

We can do it if we have first succeeded in proving a goal of the form:

```
(MODULE $?X)
```

Proof of statement (c) can be requested by asking the MICROPLANNER interpreter to process the expression (d), which asks for proof of goal (c):

```
(THGOAL (TABLE TI) (THTBF THTRUE))
```

If we had asked the MICROPLANNER interpreter to process the following expression:

```
(THGOAL (MODULE TI))
```

it would have looked in the problem-base to see whether the corresponding assertion existed and the search would have succeeded. However (TABLE TI) is not asserted and, therefore, we need a theorem for proving it. The specification (THTBF THTRUE) allows the MICROPLANNER interpreter to try, one after another, all the available theorems in the problem-base. Thus, in order to process statement (d), it finds theorem THEOREM1, proves goal (MODULE TI), and succeeds. If we had asked: "Does any module exist on the table?", the corresponding MICROPLANNER statement would have been:

```
(THPROG (Y) (THGOAL (TABLE $?Y) (THTBF THTRUE)))
```

where the THPROG has the function of an existential quantifier. When the theorem is called, the variable Y will be bounded with TI, and the answer will be:

```
(TABLE TI)
```

The MICROPLANNER interpreter can prove other goals. For instance, we can add the two following assertions to the problem base:

```
(THASSERT (MODULE EL))
(THASSERT (PIECE EL))
```

We can now ask: "Is there any piece on the table?", which corresponds to the following MICROPLANNER statement:

```
(THPROG(X)(THGOAL(TABLE $?X)(THTBF THTRUE))
          (THGOAL(PIECE $?X)) )
```

The first goal can be satisfied, as before, and the variable X is bounded to TI. But, since the assertion (PIECE TI) is not in the problem-base, the second goal fails and TI is not the piece we want. We have a backup; the MICROPLANNER interpreter comes back to the last successful goal and looks into the problem-base for a new value that can be bound to X. He finds EL, which also satisfies the second goal, and therefore the answer is the value of the second goal, namely

```
(PIECE EL)
```

MICROPLANNER theorems can be thought as subroutines, but they are called by a general procedure of pattern-matching that performs the associative search on the basis of the goals they are adapted to satisfy. The MICROPLANNER interpreter is written in LISP and has a success-and-failure mechanism capable of exploring the subgoal tree.

Ideas About the Design of Natural Language Interfaces to Query Systems

G. Guida

INTRODUCTION

The ability to converse in natural language with a data base system will enable the naive user to utilize the large set of formulated data contained in it without the necessity for specialized training. The validity of this statement requires a short discussion. The syntactic and semantic characteristics of natural language (richness, incompleteness, ambiguity, graduality) make them essentially inappropriate for communicating with computer systems having a limited set of commands and requiring precise, correct, and unambiguous inputs [1]. Therefore the situations in which the use of the natural language can be considered as valuable must have quite particular characteristics. For example, natural language interfaces would be very profitable in unstructured situations where the user is unfamiliar with the problem domain and the operative characteristics of the system. Moreover, they could be useful in interactive problem solver systems where the computer is used for processing unstructured and unfamiliar information in a goal-oriented manner.

More precisely, we can argue that the utility of natural language becomes relevant in uncertainty situations of one of the following types:

- uncertainty about the content and the use of the system;
- uncertainty about the domain of the problem.

A natural language interface will allow a user to start work on a problem in spite of these uncertainties. Such natural language interfaces are intended to bring into communication with a computer system not experienced programmers or habitual users but casual users, whose interaction with the system is irregular in time and not motivated by their job or social role [2]. Such users cannot be expected to be knowledgeable about computers, programming, computability, or logic, or to be willing to learn an artificial language, even if it is oriented towards nonprogrammers.

The purpose of this paper is to propose new design criteria for natural language interfaces to query systems. The design of such systems does not imply the implementation of a system able to understand natural language, which is in fact a very ambitious

and quite unrealistic goal at the present state of research in this field [11]. It only requires the design of a much more simple functional block that can capture and extract the operative information contained in natural language queries. The user's requests are therefore not understood in all their details and nuances, but only their operative meaning is taken into account. In this way, free use of natural language, or more precisely the illusion of free use, is allowed to the user.

It is our opinion that the design of efficient systems satisfying these conditions is not a task exceeding the present possibilities of the art. In the next section we shall present in detail the block structure of a natural language interface reflecting the basic ideas outlined above. We shall then go further into the model proposed by means of a simple design example.

BLOCK STRUCTURE

This section presents and discusses in detail the fundamental structure of a natural language interface to a data base system, and outlines the basic design criteria for each functional block in it. Let us first present some preliminary notions.

The information required for starting the design of a natural language interface to a data base system can be summarized as follows:

- information about the data base: data logical structure, access keys, information retrieval commands proper to the query language of the data base;
- information about the problem domain: natural language key words and fundamental logical links between them, possible user's requests;
- information about the natural language: basic logical-syntactic structure of the language, vocabulary with semantic interpretation of the words.

The first step toward the design of the interface is the definition of the subset of the natural language that the system must be able to understand correctly. This subset can be considered as composed, from a conceptual point of view, of two parts: a semantic block, containing a vocabulary and a collection of possible semantic links between the words, and a syntactic block, defining the basic logical structure of the sentences. The problem of defining an adequate subset is of basic importance [1], [3], [4], [5]. In fact from this subset determines the efficiency of the whole system. If this subset is too wide the parser becomes complex and inefficient; if it is too small the system meets insurmountable difficulties in understanding the user's requests and the clarification dialogue becomes heavy and meaningless.

This problem has been discussed by Watt [4] under the name of habitability. He defines the following two subsets [1]:

- adequate subset that allows one and only one way of expressing each kind of request that the user may wish to make to the system;

- fully habitable subset within which a user can carry out a dialogue about the problem domain without overstepping its bounds.

The optimal subset must be wider than the adequate subset and smaller than the fully habitable one. It is often referred to as a comfortably habitable subset [1] and its definition depends, at the present state of research, primarily on the ingeniousness, experience, and intuition of the designer. Here we emphasize the merely operative understanding capabilities of the system, and shall denote the resulting interfaces as natural oriented language systems.

Let us now present the basic structure of the natural language interface we propose (see Figure 1). Figure 2 summarizes in a flow-chart the fundamental model of operation of the system.

The first block of our model, the interpreter, receives as input the user's request and must be able to generate a symbolic expression indicating the data, the procedures, and the correct logical links between them necessary to construct a correct answer. To this end it must analyze the user's natural language request by means of a semantic-based parsing strategy [8] controlled by a logical syntactic monitor system. The first part of the analysis is semantically oriented: the words (content words and function words [8], i.e. substantives, adjectives, and verbs) are recognized, their meaning is evaluated by means of the vocabulary and the semantic templates of the system, and pointers are generated to the symbolic names of the records, record fields, and procedures stored in the data base that contains the information they denote. All words not recognized by the system, i.e. not in its vocabulary, are ignored. The concept of semantic-based parsing has already been widely discussed in [8]. We generally agree with the simple model proposed which seems to ensure good efficiency and sufficiently high understanding capabilities. Nevertheless, we believe that a syntactic check of the logical structure of the queries can make the parser much more reliable avoiding trivial misunderstanding. Hence, after the semantic-based parsing, a logical-syntactic monitor is activated which controls the logical links between the data elements discovered during the semantic analysis and connects the data items together in the correct way. It takes into account the logical connectives between the words and the propositions (conjunctions, prepositions), looks up their meaning in the vocabulary of the system, retrieves their arguments, and recognizes the correct logical structure of the sentence.

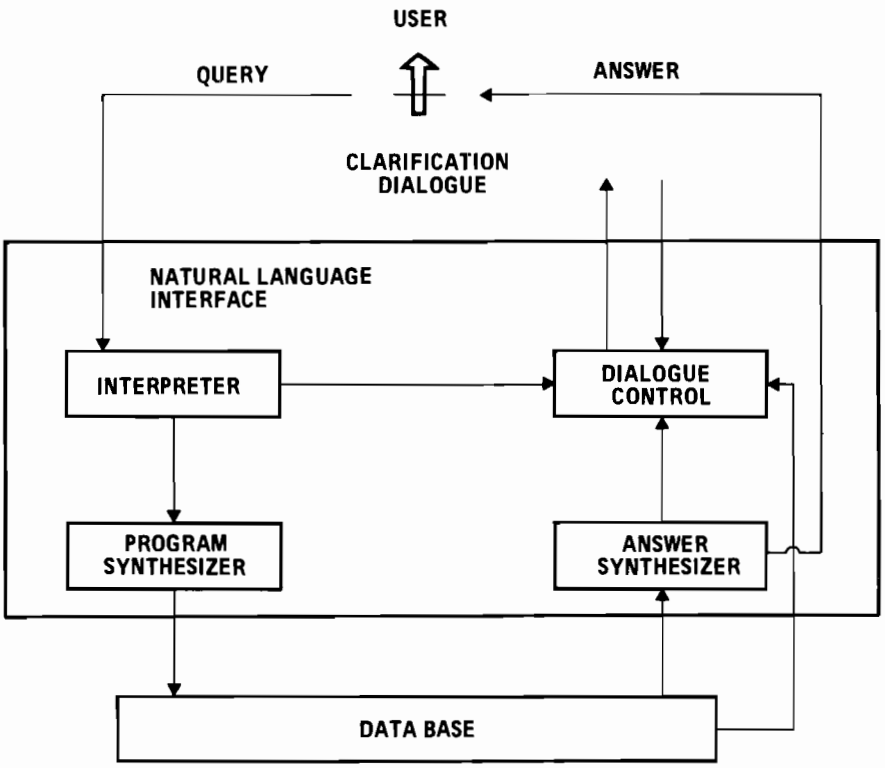


Figure 1. Block structure of a natural language interface.

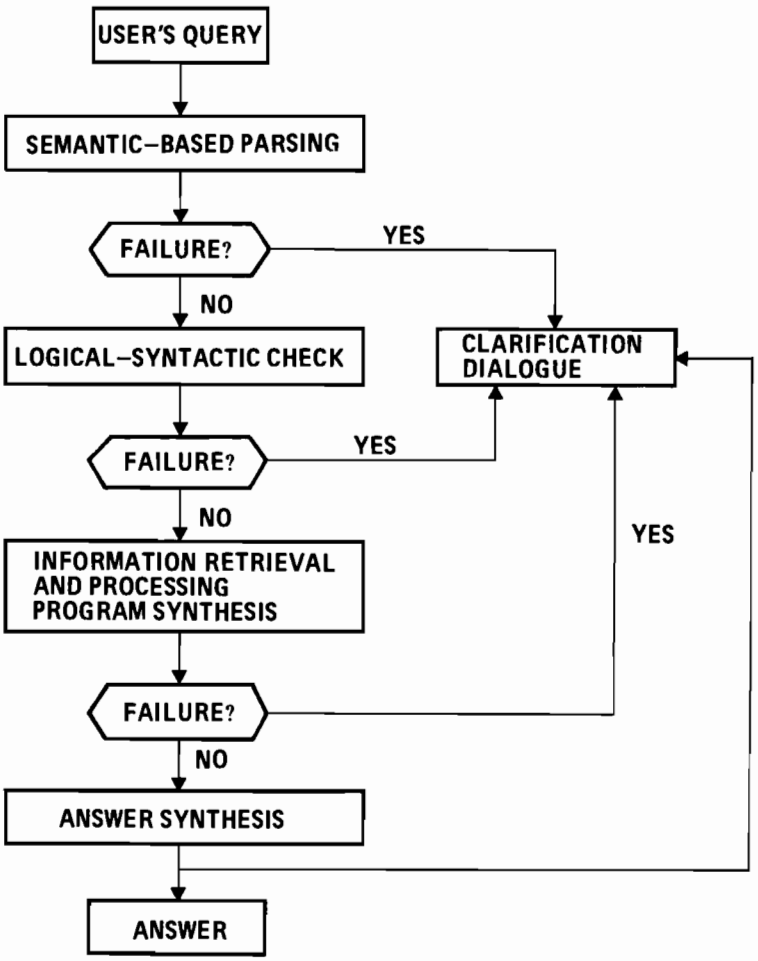


Figure 2. Mode of operation of the system.

The parsing of the user's request must allow the interpreter to identify two different types of information; what data must be retrieved in the data base, and what procedures, if any, are required for answering the request. Therefore the output of the interpreter will be an (unambiguous) expression having as arguments the symbolic names of the formatted data and of the procedures in the data base, and as connectives logical and operative symbols.

The program synthesizer is devoted to translating the symbolic expression it receives from the interpreter into appropriate information retrieval programs for the data base and to select and activate the utility programs required for processing the retrieved information. The internal structure of the program synthesizer is not much different from that of a very simple interpreter, and does not require a wide discussion. We only recall the information retrieval program generator proposed by Gerritsen [9] which utilizes typical artificial intelligence techniques, such as problem solving and goal-oriented languages (LISP-MICRO-PLANNER).

The answer synthesizer receives as input the information extracted from the data base and processed in the appropriate way and constructs the reply to the user's request. Its activity is fundamental from a functional point of view, but it is not transparent to the user. In fact, the purpose of this block is to constitute a bridge between the data base and the interface. The answer synthesizer receives from the data base the data that constitute the conceptual content of the answer and the control of the system, and activates the appropriate output routines. From the point of view of the user its activity is quite trivial; it simply adds to the results received from the data base a short illustrative sentence, chosen within a given collection.

Less deterministic and more powerful answer synthesizers can be conceived, but we do not believe that such a block is worth greater design effort.

Let us now discuss the functions of the dialogue control. This block is activated in the following situations:

- the interpreter fails in the semantic-based parsing (e.g. no word is recognized) or in the logical-syntactic check (e.g., ambiguities in the user's request cannot be eliminated by the interpreter);

- any failure occurs during the information retrieval or processing stage;

- after the answer has been communicated to the user, an incorrect matching between the answer and the user's request appears;

- after the user has communicated to the system a mismatching or a correct matching between his request and the answer received.

In each one of the above outlined situations the dialogue control receives the control of the system and starts with the user a short clarification dialogue. The clarification dialogue can request the user to express his query in a simpler or less ambiguous way, or to substitute some word, or else to answer some specific query of the system. It also communicates to the user the causes of any failure.

The activity of the dialogue control is defined in a quite deterministic way by the commands it receives. The sentences it needs for setting up the clarification dialogue can be simply chosen within a given collection, or can be constructed following some simple sentence schemata. The mode of operation of this block is controlled by a logical structure that activates, and completes if necessary, the appropriate dialogue elements in correspondence to the different commands received.

After the clarification dialogue is terminated the dialogue control returns the control to the appropriate block of the system.

The learner should be able, first of all, to take advantage of the past experience of the system and, hence, of storing in an appropriate way the most significant elementary components of its activity. Then, it should apply this information for dynamically improving the capabilities and the behavior of the system. An efficient way to obtain this result is, in our opinion, to allow the learner to manage in a dynamic way the vocabulary of the system, acting both on its ordering and on its content. The new information the learner needs for such a purpose can be obtained from an appropriate clarification dialogue, which represents therefore the most important source of experience for the system. However, our basic point of view is that simple prototypes of natural language interfaces do not require such high level capabilities. Only when the implementation of the block structure proposed has reached a good level of efficiency and low cost can the design of an additional learner be started. For this reason we have omitted such a block in our model (Figure 1).

A DESIGN EXAMPLE

In this section we go further into the model proposed by means of a simple design example, the purpose of which is to allow the reader to critically evaluate the relative importance of the different functional blocks of the model, and to determine the crucial points of the design. Moreover, the simple solution proposed can constitute a helpful base to develop more advanced and efficient design techniques. We shall focus our attention on the design of the interpreter which is, in our opinion the heart

of the whole system. For the other blocks, which have a quite simple logical function and the relevance of which is primarily bound to implementation reasons, only elementary design examples will be proposed.

Let us assume that the user's request is the following one:

> I WOULD LIKE TO KNOW HOW MANY STUDENTS IN THE DEPARTMENT OF MATHEMATICS ARE AGED OVER 25 YEARS.

The first step of the interpreter activity is the semantic-based parsing. The interpreter scans the user's request in order to recognize the content words and the function words contained in it. For each word in the request the interpreter executes first a searching in the content words part of the vocabulary of the system.

The content words part is a collection of couples (X,Y) such that:

- X is a list of content words, i.e. of natural language words referring to a particular record or record field, or to the content of a particular record field;
- Y is the symbolic name of the record (RY) or record field (FY) to which the words of X refer, or if X denotes the content of a record field, the symbolic name of the record field (CFY) in which X can be contained.

Examples of such couples are:

```
((DEPARTMENT(S)), FDEP)
((LOGIC, MATHEMATICS), CFDEP)
((LOGIC, MATHEMATICS), CFEXAM)
((STUDENT(S)),RSTUD)
((AGE(D)), YEAR(S)), FAGE)
```

Please note the particular use of the parentheses inside a content word to denote several possible forms of a given word.

Whenever a content word is recognized the Y part of the couple in which it appears is bound to it. If it appears in more than one couple (in our example the word MATHEMATICS) all the appropriate Y's are bound to it. If no content word is recognized the dialogue control is activated (flag 1, see Table 1).

In our example the content words recognized are:

```
STUDENTS     → RSTUD
DEPARTMENT   → FDEP
```

Table 1. Dialogue control.

flag - parameters	clarification dialogue	control
1 -	I CAN NOT UNDERSTAND YOUR QUESTION WOULD YOU RESTATE IT IN A DIFFERENT WAY?	interpreter
2 - $X_1,X_2,\ldots,X_n$	I CAN NOT UNDERSTAND THE WORDS $X_1,X_2,\ldots,X_n$ CAN YOU RESTATE YOUR QUESTION?	interpreter
3 - $X_1,X_2,\ldots,X_n$	THE WORDS $X_1,X_2,\ldots,X_n$ CAUSE AMBIGUOUS INTERPRETATION CAN YOU RESTATE YOUR QUESTION?	interpreter
4 - $X_1,X_2,\ldots,X_n$	I CAN NOT CORRECTLY CONNECT TOGETHER THE MEANING OF THE WORDS $X_1,X_2,\ldots,X_n$ CAN YOU RESTATE YOUR QUESTION?	interpreter
5 - 6 -	I CAN NOT RETRIEVE IN THE DATA BASE THE INFORMATION NEEDED TO ANSWER YOUR QUESTION	interpreter
7 -	DOES THE ANSWER OBTAINED MATCH CORRECTLY WITH YOUR QUESTION? PLEASE REPLY YES OR NO ONLY	dialogue control
8 -	SYSTEM READY FOR OPERATION	interpreter
9 -	YOU MAY TRY TO RESTATE YOUR QUESTION IN A DIFFERENT WAY	interpreter

```
MATHEMATICS → CFDEP, CFEXAM
AGED        → FAGE
YEARS       → FAGE
```

After all the words in the user's request have been tested for being content words, the interpreter starts another similar search in the function words part of the vocabulary.

The function words part is a collection of couples $(X,Y\ (D_1,D_2,\ldots,D_n - R_1,R_2,\ldots,R_m))$ such that:

- X is a list of function words, i.e. of natural language words referring to a particular processing the system is able to perform on the information contained in the data base;
- Y is the symbolic name of the particular processing to which the words of X refer;
- $D_1,D_2,\ldots,D_n$ denote the number, the type, and the correct correspondence of the input paramters, i.e. the data, of the process Y;
- $R_1,R_2,\ldots,R_m$ denote the number, the type, and the correct correspondence of the output parameters, i.e. the results, of the process Y.

An example is:

```
((HOW MANY, THE NUMBER OF) , (NUM(RECLIST, INUM)))
```

Please note that a function word can also be constituted by a sequence of words. Whenever a function word is recognized the $Y(D_1,D_2,\ldots,D_n - R_1,R_2,\ldots,R_m)$ part of each couple in which it appears is bound to it.

After these searching activities are terminated, if any function word has been recognized the interpreter starts a matching process. For each function word it verifies whether content words which have been recognized can constitute correct input parameters for the function word considered. Each input parameter is bound to all the content words of the correct type. If some parameter remains unbound the dialogue control is activated (flag 2), and the function words $X_1,X_2,\ldots,X_n$ with unbound parameters are communicated to the dialogue control.

In our example the only function word recognized is HOW MANY and its input parameter RECLIST (list of records) is bound to the only content word of type record STUDENT.

At this point the second step of the interpreter activity, the logical-syntactic check, begins. The user's request is scanned once again in order to discover the logical-syntactic connectives contained in it. To this purpose the interpreter executes a searching in the connectives part of the vocabulary.

The connective part is a collection of couples (X,Y) where:

- X is a list of connectives, i.e. of natural language words which denote a particular syntactic or logical relation between content words,
- Y is a symbolic expression denoting the meaning of the connectives X and, if it is the case, the number, type, and correct correspondence of their arguments.

Examples of connectives are:

```
((OVER), GT (-NUMBER))
((OF), (FIELDY, CONTENTY))
((IN), (RECORD, FIELD))
```

The purpose of the logical-syntactic check is to connect in the correct way the content words and to eliminate any ambiguous bond occurring in the preceding stages of the parsing. In our example the connective OF allows the removal of the only ambiguous binding.

```
MATHEMATICS → CFDEP, CFEXAM
```

to

```
MATHEMATICS → CFDEP
```

Moreover, the connectives IN and OVER allow all the content words recognized to be connected in the correct way.

If the logical-syntactic check fails in the connection of the content words or in the elimination of the ambiguities the dialogue control is activated (flag 3 and flag 4 respectively), and the words $X_1, X_2, \ldots, X_n$ with ambiguous bonds or which cannot be correctly connected are communicated to the dialogue control. All the words in the user's request not recognized as content words, function words, or connectives, and not numbers, are ignored.

Before proceeding further let us outline the fundamental importance of the vocabulary in the design of the whole system. From it depends primarily the efficiency and the understanding capability of the interface. Its definition is a critical point of the design. The correct way for defining the vocabulary should be an incremental process, which, starting from a small set of words assigned a priori, gradually modifies its content and structure during an appropriate testing stage of the system. Moreover, such a process should be considered as a first step toward the definition of learning strategies.

To complete its parsing activity the interpreter needs three more tables:

- a record table, in which for any symbolic name of a record field the symbolic name of the record to which it belongs is indicated;
- a file table, in which for any symbolic name of a record the file in which it is stored and the retrieval keys are indicated;
- a function table, which specifies for any symbolic name of a particular processing bond to a function word, the library programs that perform the processing and the correct way in which they must be activated.

Examples of such tables are:

DEP	STUD
AGE	STUD
NAME	STUD

STUD	FILEA	NAME

NUM	PROG 1	RECLIST → X1, INUM → X2

The interpreter is now able to generate the correct parsing of the initial user's request, which is a symbolic expression of the following type:

```
RETRIEVE IN FILEA ON NAME
STUD (DEP=MATHEMATICS, AGE GT 25)
EXECUTE PROG 1(X1 = STUDLIST, X2)
RETURN X2
```

The above expression is then supplied as input to the program synthesizer. This block translates the expression it receives into a program in the query language proper to the data base to which it is connected. Its activity is quite simple since it must only insert the appropriate parameters into a parametric program model.

An example of such a model for a simple query language could be of the following type:

```
OPEN FILE X
RETRIEVE XR1,XR2,...,XRN ON K1,K2,...KN
CLOSE FILE X

ACTIVATE SUBROUTINE S' (Y1,Y2,...,YM)
EXECUTE S(Y1',Y2',...,YM')

RETURN DATA AND CONTROL
```

In our case we would have:

```
OPEN FILE A
RETRIEVE STUD (DEP = MATHEMATICS, AGE GT 25) ON NAME
CLOSE FILE A
ACTIVATE SUBROUTINE PROG1 (Y1, Y2)
EXECUTE PROG1 (STUDLIST, X2)
RETURN X2 AND CONTROL
```

After this program has been generated control is assumed by the base management system which executes it. If any failure occurs during the execution of the information retrieval or processing programs the control is returned to the interface and the dialogue control is activated (flag 5 and flag 6 respectively).

When the control and the data return to the interface the answer synthesizer is activated. It simply communicates to the user the results obtained and possibly, it adds to the data an illustrative sentence chosen, at random or according to some control signal generated by the interpreter, in a given list. In our example a simple standard answer could be:

```
THE ANSWER TO YOUR QUESTION IS ...
```

After the answer has been generated the answer synthesizer activates the dialogue control (flag 7), which asks the user about the correct matching between his request and the answer obtained. If the user communicates a correct matching (YES - flag 8) the interface concludes its activity, the control returns to the interpreter, and the system is ready to examine another request. Otherwise, the dialogue control is activated once again (NO - flag 9).

The activity of the dialogue control can be defined in a simple way by means of a deterministic table. A simple example of this is presented in Table 1.

CONCLUSIONS

Implementation problems have not been directly taken into account in this paper. The model proposed is implicitly oriented toward a software implementation. It could be appropriate for interfaces devoted to connect users with large data base systems implemented on big computers. The most relevant problems related to software implementation are choice of the appropriate language and management of the control transfers between the interface, the data base management system, and the operating system of the computer. Artificial intelligence languages, such as LISP or MICROPLANNER, which can be usefully adopted for the design of simple prototypes, are not valuable for the implementation of real systems.

Hardware implementations of the model proposed are also possible and constitute a promising and stimulating research direction. Such an implementation would allow the design of a natural language interface with a fully independent and modular peripheral unit composed of three blocks:

- a core block, comprising the interpreter, the answer synthesizer, and the dialogue control;
- a vocabulary, devoted to tune the system into different semantic domains;
- a data base interface, comprising the program synthesizer, devoted to connect the system to different computer and data base management systems.

Many problems have been raised and left unanswered. Examples are the definition criteria for the comfortably habitable subset and for the vocabulary, the techniques for eliminating the parsing ambiguities, and the choice of an adequate intermediate language for expressing the output of the interpreter. Nevertheless, the notions presented should be adequate for experienced programmers to visualize several approaches to the design of prototypes and of real systems. Moreover, we hope that the model proposed can show clearly the new idea on which it is based and can contribute to further progress in this field.

REFERENCES

[1] Malhotra A., and I. Wladawsy, *The Utility of Natural Language Systems*, IBM Research RC 5739 Computer Sciences, IBM, White Plains, New York, 1975.

[2] Codd, E.F., *Seven Steps to Rendezvous with the Casual User*, IBM Research RJ 1333, IBM, White Plains, New York, 1974.

[3] Montgomery, C.A., Is Natural Language an Unnatural Query Language?, *Proceedings of an ACM National Conference*, Association of Computing Machinery, New York, New York, 1972.

[4] Watt, W.C., Habitability, *American Documentation*, 19, 3 (1968).

[5] Malhotra, A., *Design Requirements for a Knowledge-Based English Language System for Management: An Experimental Analysis*, TR 146 Project MAC, MIT, Cambridge, Mass., 1975.

[6] Guida, G., Tecniche di Intelligenza Artificiale per il Progetto di Banche di Dati, in *Proceedings Annual AICA Conference*, Milan, October 1976.

[7] Andrew, A.M., Artificial Learning Systems and QAS, in *Conference on Artificial Intelligence: Question-Answering Systems*, CP-76-6, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1975.

[8] Burger, J., A. Leal, and A. Shoshani, Semantic-Based Parsing and a Natural Language Interface for Interactive Data Management, in *Proceedings 13th Annual Conference of the Association for Computational Linguistics*, Boston, 1975.

[9] Gerritsen, R., *The Application of Artificial Intelligence to Data Base Management*, presented at the 4th IJCAI, Tbilisi, August, 1975.

[10] Antonacci, F., P. Dell'Orco, and V.N. Spadavecchia, *An Artificial Intelligence Approach to the Semantic Evaluation of Natural Language Queries*, IBM Italia, Bari Scientific Center, Bari.

[11] Crespi Reghizzi, S., and D. Mandrioli, *Basi, Problemi e Prospettive della Elaborazione Automatica del Linguaggio Naturale*, Rapporto interno 76-10, Istituto di Elettrotecnica ed Elettronica del Politecnico di Milano, Laboratorio di Calcolatori, Milan 1976.

Two Paradigms for Natural Language And Data Bases*

R. Stamper

I should like to draw attention to the way in which we have been using the expression "natural language" during this meeting. Our usage will seem quite appropriate to members of the intellectual communities concerned with artificial intelligence and computational linguistics. However, I propose that for business users of data bases, "formal languages with structures approximating to those of natural language" would be a more correct description. This other intellectual community has a different set of problems to solve. The difference of viewpoint can be explained by indicating the two paradigms that might be employed when studying the design of data base systems.

The paradigm, for the study of natural language for interaction with data bases, implicit in all the papers presented at this meeting, might be depicted by Figure 1.

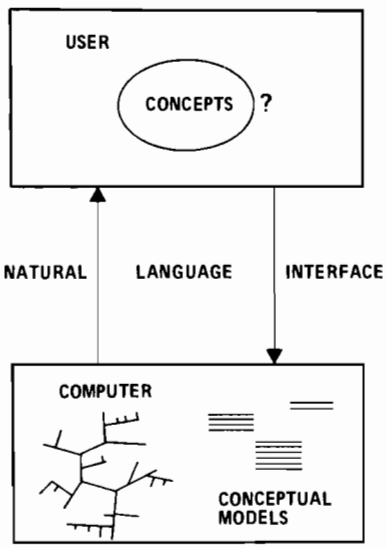


Figure 1. Paradigm for computational linguistics or artificial intelligence.

*Report of an informal presentation.

The empirical methods suggested by this model are directed towards discovering structures in the language that users would like to employ when communicating with a data base. One aim is to identify the concepts needed. The individual is presumed to arrange these concepts mentally in ways that can be modeled by network, relational, or other structures within a computer. The interface is the "natural language" we have been talking about at this Workshop.

This way of using the words "natural language" is apt in the context of our present discussions which have taken place from a point-of-view of linguistics and artificial intelligence. The quest has been for structures in language utterances that might be related to computationally convenient structures in a data base. The result is to devise some specialized languages that, for the benefit of the user, approximate natural language structurally. The limitations of these languages must not be overlooked lest we apply the results inappropriately. For the typical business user of data bases it might be better to admit that they are formal languages.

There is another paradigm better fitted to business systems design. This places the data base language problem in a different context and uses the expression "natural language" differently. Figure 2 depicts this model. It distinguishes three system levels:

- the real world in which we focus upon a limited subset of things, the object system, relevant to some practical problem;

- natural language can then be used to attend to this practical problem and it is employed by people in an informal way within the discourse system;

- if necessary, we can add a formal system in which signs are used according to explicit rules and definitions.

In this second paradigm there is a profound distinction between formal and natural language. Natural language exists by virtue of human social behavior; within the discourse system, signs, including language signs, acquire their meanings as social norms; explicit definitions are not necessary.

Formal systems are added for two major reasons. For performing some large-scale tasks, the discourse system is defective because informal norms are not easily imposed uniformly on a large community for a long span of time. In such circumstances, special words and usages can be supplied by explicit definitions and rules (as in administrative law). We were familiar with formal systems for this first reason before the computer's advent. People could be employed, typically as clerks, to manipulate words and numbers in mechanical ways according to formal rules. Today, we can employ computers for this work and, because computers are unable to acquire socio-linguistic norms, we can employ them only in formal

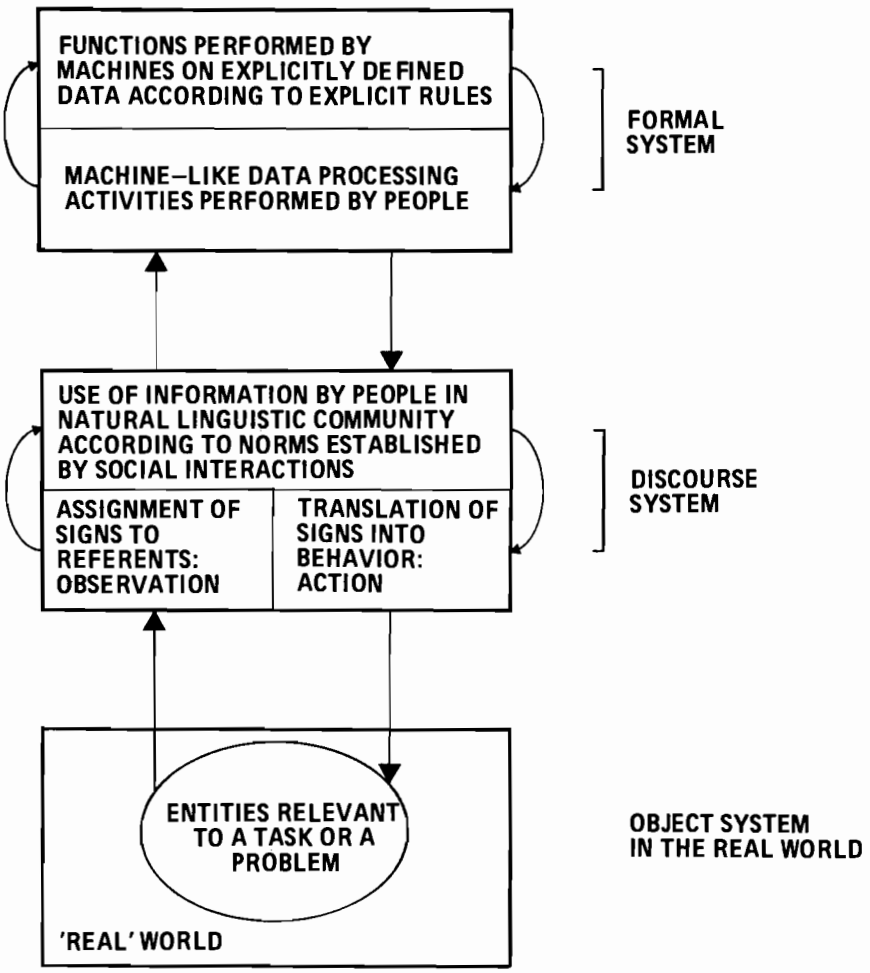


Figure 2. Paradigm for practical data processing.

systems. This is the second major reason for adding formal systems to our natural ones; to enable us to exploit our new information technology. The difference between the two paradigms is not merely one of terminology: the "natural language" of paradigm 1 being replaced by "a special class of formal language" in paradigm 2. It is more significant that problems of semantics are approached differently according to the paradigm one adopts. With paradigm 1, meanings are explicated structurally. An example that arose in Solovitz's presentation was the expression HOT DOG* which has two different meanings depending on whether it is regarded as a species of canine or a species of sandwich. These may be represented by appending the same words to different parts of a semantic structure: HOT DOG as a unary expression will be the idiomatic name for a kind of sandwich; HOT will also be a permitted description or specializer for DOG. This type of analysis may be arrived at by asking how HOT DOG, HOT, and DOG are used in the realm of natural language. Interface and storage structures will then be constructed appropriately to embody the meanings of these expressions.

With paradigm 2, notions of meanings are almost literally brought down to earth. One cannot say what words mean within the discourse system by giving formal definitions or logical structures; that would be to retreat to an abstract level of analysis away from the tangible world of the practical problem-solver. To make him confident that words and numbers are being correctly used, one has to demonstrate meanings ostensively and show that signs are effective tools for solving practical problems. These are problems with well-defined outcomes and if language is to be necessary, they involve people in solving them collectively. The meanings of words depend upon the contexts established by these problems, the actions to be performed, and the results desired. In a greyhound-racing stadium, for example, a person asking for a "hot dog" would be sent to the cafeteria to assuage his hunger, to the kennels to treat a sick animal, and to a tipster to find a competitor worth betting on.

To resolve disputes about meaning in paradigm 1 we may employ a competent user of the language to arbitrate. In paradigm 2, however, the criteria for resolving semantic disputes are objective tests of whether the meaning is suitable for performing some *task*, solving some *problem*, or attaining some *goal*. Even with the simplest problem of data base semantics, the identification of physical objects [1], paradigm 2 leads to multiple solutions that would not readily be distinguished in paradigm 1 by a competent user of the language on grounds of purely linguistic knowledge. For example, appearances may be deceptive as when a body continues its physical existence whilst the existence of a person ends and the existence of a corpse begins; modern medical techniques have removed the relative clarity of this change.

---

*This example was used in the oral presentation by P. Solovitz but is not used in the paper by Solovitz et al. in these proceedings.

Alternatively, we ignore the physical transmutation of, say, a production line, although it has been totally replaced part by part, and treat it as the same object because of its systemic continuity. These semantic problems must be resolved by the information analyst working on a business application: he is the person responsible for choosing the appropriate "myth of physical objects" (to use Quine's expression). The myth is "real" enough only if it serves to get a job done correctly.

The two paradigms are not competitors. They are both relevant for understanding and designing data base systems but only paradigm 2 is necessary. The correct operational links between data and the things they represent are not merely desirable but indispensable if the data base is to generate the correct organizational behavior. On the other hand, an interface with linguistic properties that assist the user may be highly desirable but it is not essential. Surprisingly, little research has yet been done on the more important problem of the operational semantics of paradigm 2.

The same linguistic solution may serve the interface designer for many different operational problems. In this respect, the two paradigms lead to techniques that can be used cooperatively. The danger is that conceptual models based upon the linguistic paradigm 2, which correspond to the linguistic norms of different subsets of the user population who are trying to solve different problems but are using the same words. One way of handling these variants was described by Stonebraker whose INGRES system permits control of VIEW, ACCESS, and INTEGRITY; these functions supply the kind of structure necessary for avoiding confusion among operational meanings.

The two paradigms are also mutually helpful in another way. Paradigm 2 leads one to think in terms of contexts distinguished by the purpose for which the data are being used. A system that does not permit the ambiguity of purpose suggested by the risk of confusing "overheated canine" with "sausage sandwich" would eliminate the linguistic problem of HOT DOG.

If they allow us to tackle effectively two smaller sets of problems, then two paradigms are better than one. The error would be to adopt one exclusively, ignoring the other. A devotee of the linguistic paradigm 1, who is trying to simulate such marks of intelligence as the disambiguation of expressions, may be tempted to ignore the pedestrian solutions of the operational analyst. The latter's narrow definition of the purpose for which the data base is used, may eliminate many fascinating problems of disambiguation with which an intelligent person can deal. Frankly, I am in favor of ignoring such fascinating problems of individual artificial intelligence if we can direct our energies toward the equally fascinating, far more urgent but overlooked problems of building a shared, public intelligence. This seems to me to be the main practical purpose of data base technology. It leads us in another direction. Perhaps another workshop could point the way.

REFERENCE

[1] Stamper, R.K., Physical Objects, Human Discourse and Formal Systems, in *Proceedings of IFIP TC2 Working Conference*, January, 1977.

APPENDIXES

Appendix A

The IIASA Energy Resources Sample Data Base

The IIASA energy resource sample data base was a very simple and partial representation of various resources and resource references. It was a simple relational data base consisting of eleven relations. These along with the field names, some sample values, and a "natural" language explanation are given in Tables 1-11.

Table 1. RCOUNTRY--Identification of countries.

COUNTRY*	COUNTRYN	ALPHA2	PMREG	Field Name
I4 2b	A24 2b	A2 2b	I2	Fortran format code (b = blank)
40	AFGHANISTAN	AF	9	
101	ALBANIA	AL	5	
1	ALGERIA	DZ	7	

*Key underlined

THE COUNTRY AFGHANISTAN IS IDENTIFIED EITHER BY THE NUMBER 40 OR BY THE CODE AF AND IS LOCATED IN THE PM REGION 9 (PESTEL-MESAROVIC).

Table 2. RPRODDEP--Production by deposits.

DEPO NO	YEAR	RES NO	QUANTITY	CUMQUANT	WELLS	REF	55 columns
I7 5b	I4 5b	I1 4b	F7.1 2b	F7.1 2b	I7. 2b	I2	
1	1973	1	293	1393	63	1	
1001	1973	2	91	179	-1*	1	
1002	1973	2	109	436	5	1	

*-1 = no data

REFERENCE 1 REPORTS THAT IN THE YEAR 1973 AN AMOUNT OF 293 UNITS OF RESOURCE 1 WERE PRODUCED USING 63 WELLS. THE CUMULATED PRODUCTION UP TO THE YEAR 1973 WAS 1393 UNITS OF RESOURCE 1.

Table 3. RRESERVE--Characterization of reserves.

DEPO NO	DEPONAME	COUN-TRY	RES NO	RESERVE	PAY	DEPTH	API	DISC DATE	REF	80 columns
I4 ↂ	A20 ↂ	I3 ↂ	I1 ↂ	F6.0 ↂ	A20 ↂ	F6.1 ↂ	F5.1ↂ	I4 ↂ	I1	
1	MURBAN BAB	186	1	1733	LK	8600	38.	1960	1	
1002	HASSI R'MEL	1	2	53564	TRIASSIC	2072.6	-1	1956	1	

-1 = No Data
(-0.3) Available
(-1.0)

DEPOSIT NUMBER 1, CALLED "MURBAN BAB", IS LOCATED IN COUNTRY 186 AND CONTAINS AN ESTIMATED RESERVE OF 1733 UNITS OF RESOURCE NO. 1 (OIL) OF AN API GRAVITY OF 38 DEGREES IN A MEDIUM DEPTH OF 8600 FEET, GEOLOGICALLY CHARACTERIZED AS "LK". THE DEPOSIT WAS DISCOVERED IN THE YEAR 1960. THE DATA REFERENCE IS 1.

Table 4. RRESOURC.

RES NO	RES NAME	QUANT DIM	
I2 2ↂ	A20 2ↂ	A10	
1	CRUDE OIL	MEGA BL	36 columns
62	WATER, INTAKE	MEGA CUM	

EXAMPLE: THE RESOURCE "CRUDE OIL" IS IDENTIFIED BY THE RESOURCE NUMBER 1 AND EVERY QUANTITY OF THE RESOURCE IS GIVEN IN MEGA BL (MILLIONS OF BARRELS).

Table 5. Identification and definition of facilities--RFAC.

FAC NO	FAC NAME	FAC CLASS	ACT	CAP QUANT	CAP UNIT	IN RES NO	IN RES QT	OUT RES NO	OUT RES QT
I3 ↳	A45 ↳	A1 ↳	4A ↳	I4 ↳	A2 ↳	I2 ↳	I4 ↳	I1 ↳	F5.3
101	RESIDUAL FUEL OIL POWER PLANT	P	EPGE	800	MW	29	1578	5	5.256
102	COAL FIRED POWER PLANT WITH LIME SCRUBBER FLUE GAS DESULFURIZATION	P	EPGE	800	MW	29	1626	5	5.256
103	LIGHT WATER REACTOR	L	EPGE	1100	MW	42	2582	5	7.227

EXAMPLE: THE FACILITY "RESIDUAL FUEL OIL POWER PLANT", IDENTIFIED BY NUMBER 101, IS A P-CLASS FACILITY (POWER PLANT), BELONGS TO THE ACTIVITY EPGE (ELECTRIC POWER GENERATION), HAS A CAPACITY OF 800 MEGAWATTS, NEEDS 1578 UNITS OF RESOURCE 29 (COAL) TO PRODUCE 5.256 UNITS OF RESOURCE 5 (ELECTRICITY).

Table 6. RWATERPC.

LAST YEAR	COUNTRY	AVPRECIP [MM/y]	AVRUNOFF [MM/y]	INRIVER [CUKM/y]	OUTRIVER [CUKM/y]	REF
I4 ↳	I3 ↳	F5.3 ↳	I3 ↳	I2 ↳	I2 ↳	I1
1967	101	1.200	350	3	13	3
1966	63	1.191	661	35	90	3
1966	64	850	360	5	16	3

THE AVERAGE PRECIPITATION OF 1.200 MM/YEAR AND AVERAGE RUNOFF OF 350 MM/YEAR AND AVERAGE FLOW OF RIVERS 3KM^3/Y RECEIVED FROM COUNTRIES SITUATED UPSTREAM AND THE AV. FLOW OF RIVERS LEAVING THE COUNTRY 13KM^3/Y IS REPORTED BY REF. 3 FOR COUNTRY 101 USING MEASUREMENTS UP TO THE YEAR 1967.

Table 7. Construction time and lifetime of facilities--RCLTIME.

FAC NO	CONTIME	REF	LIFETIME	REF
I3 1\5	I1 1\5	I2 1\5	I2 1\5	I2 1\5
101	5	10	30	10
102	5	10	30	10
103	9	10	30	10

FACILITY 101 HAS A LIFETIME OF 30 YEARS AND A CONSTRUCTION TIME OF 5 YEARS. SEE REFERENCE 10

Table 8. Required resources for the operation of facilities--ROPERREQ

FAC NO	RES NO	RESQUANT	REF
I3 1\5	I2 1\5	F7.3 1\5	I2
103	60	18.975	10
103	70	1.207	10
103	71	0.207	10
103	52	112	10
101	60	8.16	10
101	70	1.080	10
101	52	84	10

18.975 UNITS OF RESOURCE 60 ARE REQUIRED TO OPERATE THE FACILITY 103 ACCORDING TO REFERENCE 10.

Table 9. Construction requirements for facilities--RCONREQ.

FAC NO	RES NO	RESQUANT	REF
I3 1\5	I2 1\5	F10.3 1\5	I2 1\5
101	52	4400	10
102	52	5250	10
103	52	12000	10
101	70	1.08	10

ACCORDING TO REFERENCE 10, ONE NEEDS 4400 UNITS OF RESOURCE 52 TO CONSTRUCT THE ENERGY FACILITY 101.

Table 10. Publications used as a data source--RPUBLICA.

IIASANO A5	AUTHOR A16	TITLE A29	PUBLISH A23	YEAR I4	PUBLNO I1
B2879	MCCASLIN, JOHN C.	INTERN. PETRO-LEUM ENCYCL.	PETR. PUBLIC. CO.	1974	1
-1	CARASSO, M	ENERGY SUPPLY PLANN. MODEL	BECHTEL CO. S.F.	1975	2

A PUBLICATION WITH THE IIASA LIBRARY ACCESSION NUMBER B2879 WRITTEN (OR EDITED) BY JOHN C. MCCASLIN HAS THE TITLE "INTERNAT. PETROLEUM ENCYCLOPEDIA", WAS PUBLISHED BY "PETROLEUM PUBLISHING COMPANY" IN 1974 AND IS IDENTIFIED BY NUMBER 1.

Table 11. RREFERNC--Data reference within a publication.

REF NO I2	PUBLIC NO I1	STARTPAG I3	COMMENT, (PRIMARY DATA SOURCE) A54
1	1	216	MAJOR OIL FIELDS AROUND THE WORLD
10	8	-1	
3	7	-1	PRIMARY DATA SOURCE: ECONOM. COMMISSION FOR EUROPE 1967

Example: REFERENCE NO. 3 IS AN INFORMATION SOURCE SPECIFICATION DEFINING A PUBLICATION WITH THE NUMBER 7 AND A PAGE NUMBER WITHIN THIS PUBLICATION. INDIVIDUAL COMMENTS ARE ASSOCIATED.

Appendix B

List of Participants

Prof. W. Abraham
Fakultiet der Letteren
Rijksuniversiteit
Groningen
Grote Kruisstraat 21
NETHERLANDS

Prof. H. Andersin
Helsinki University of
Technology
Otaniemi
Espoo
FINLAND

Dr. F. Antonacci
IBM Centro di Ricerca
Via Cardassi 3
Bari
ITALY

Dr. J. Bankowski
Institute for Scientific
Technical and Economic
Information
Warsaw
POLAND

Dr. G. Berry-Rogghe
Institut für deutsche Sprache
Friedrich-Karl-Strasse 12
Mannheim
FRG

Dr. S. Braun
Institut für Informatik
Technische Hochschule München
Arcisstrasse 21
Munich
FRG

Dr. V. Briabrin
Computing Center
Academy of Sciences
40 Vavilova Street
Moscow
USSR

Dipl. Ing. M. Burghardt
Siemens AG
D AP GE
Boschetsriederstr. 41
Munich
FRG

Dr. A. Butrimenko
IIASA
Schloss Laxenburg
Laxenburg
AUSTRIA

Dr. V. Cherniawsky
Technische Universität Berlin
Berlin
FRG

Dr. Dehtiarenko
Minsk Institute of Technical
Cybernetics
Minsk
USSR

Dr. P. Dell'Orco
IBM Italia SPA
Centro di Ricerca
Via Cadassi
Bari
ITALY

Dr. M.J. Ferguson
IIASA
Schloss Laxenburg
Laxenburg
AUSTRIA

Dr. G. Gell
Universitätsklinik für Radiologie
Auenbruggerplatz 9
Graz
AUSTRIA

Dipl. Math. G. Goerz
Universität Erlangen-Nürnberg
Rechenzentrum
Martensstrasse 1
Erlangen
FRG

Dr. G. Guckler
Seminar f. Allg. u. Vgl.
Sprachwissenschaft
Postfach 3980
Mainz
FRG

Dr. G. Hendrix
Stanford Research Institute
333 Ravenswood Ave.
Menlo Park, California
USA

Dr. O. Itzinger
Institute for Advanced Studies
Stumpergasse 56
Vienna
AUSTRIA

Ms. M. King
Istituto Per Gli Studi
Semantici e Cognitivi
Fondazione, Dalle Molle
Université de Geneve
17 Rue De Candolle
Geneva
SWITZERLAND

Mr. D. Kolb
Institut für deutsche Sprache
Abt. IDV
Friedrich-Karl-Strasse 12
Mannheim
FRG

Dr. K.-D. Krägeloh
Universität Karlsruhe
Institut für Informatik II
Karlsruhe
FRG

Dr. G. Lau
Österreichische Computer GesmbH
Schottengasse 3/1
Vienna
AUSTRIA

Dr. E. Lehmann
Academy of Sciences of the GDR
Rudower Chaussee 5
Berlin
GDR

Dr. H. Lehmann
IBM Wissenschaftliches Zentrum
Tiergartenstrasse 15
Heidelberg
FRG

Phys. M.S. G. Liebisch
Industrieanlagen-
Betriebsgesellschaft mbH
Einsteinstrasse
Ottobrun
FRG

Dr. Mikulich
Moscow Institute of Control
Sciences
Moscow
USSR

Dr. G. Marini
Systems Engineer
IBM Scientific Center
Dorsoduro 3228
Venice
ITALY

Prof. H.W. Meier
Academy of Sciences of the GDR
Rudower Chaussee 5
Berlin
GDR

Dr. D.A. Pospelov
Computing Center
Academy of Sciences
40 Vavilova Street
Moscow
USSR

Dr. G.S. Pospelov
Computing Center
Academy of Sciences
40 Vavilova Street
Moscow
USSR

Dr. W. Rauch
Institut für Sozio-Ökonomische Entwicklungsforschung
Fleischmarkt 20
Vienna
AUSTRIA

Dipl. Ing. G. Rahmstorf
IIASA
Schloss Laxenburg
Laxenburg
AUSTRIA

Prof. P. Rivera
Politecnico di Milano
Pizza Leonardo da Vinci 32
Milan
ITALY

Doz. Dr. L. Reisinger
Institut für Statistik der Universität Wien
Rathausstrasse 19/4
Vienna
AUSTRIA

Dr. Schubert
Academy of Sciences of the GDR
Zentralinstitut für Kybernetik und Informationsprozesse
Rudower Chaussee 5
Berlin
GDR

Prof. M. Somalivico
Politecnico de Milano
Milan Polytechnic Artificial Intelligence Project
Piazza Leonardo da Vinci n 32
Milan
ITALY

Mr. R. Stamper
London School of Economics and Political Science
Houghton Street
London
UK

Dipl. Ing. P. Staudigl
Technical University of Vienna
Dept. Organization and Computer
Karlsplatz 13
Vienna
AUSTRIA

Prof. M. Stonebraker
College of Engineering
Dept. of Electrical Engineering and Computer Science
University of California
Berkeley, California
USA

Dr. J. Sturc
Computing Research Center
Bratislava
CZECHOSLOVAKIA

Dr. Subieta
Computer Center
Academy of Sciences
POB 22
Warsaw
POLAND

Dr. P. Szolovits
MIT Laboratory for Computer Sciences
545 Technology Square
Cambridge, Massachusetts
USA

Dr. E. Tyugy
Tallinn Institute of Cybernetics
Tallinn
USSR

Mr. D. Vargha
Computing Center of the Hungarian Planning Office
Angol u. 27
Budapest
HUNGARY

Dr. Wang
Technische Universität Berlin
Berlin
FRG

Mr. E. Warman
Perkins Engines Co. Ltd.,
Frank Perkins Way
Peterborough
UK

Prof. A. Zampolli
Linguistics Division
Istituto Del Consiglio Nazionale Delle Ricerche
CNUCE
36 Via St. Maria
Pisa
ITALY