SRI International

NATURAL-LANGUAGE INTERFACES

Technical Note 393

August 22, 1986

By: C. Raymond Perrault and Barbara J. Grosz
Artificial Intelligence Center
Computer Science and Technology Division
and
Center for the Study of Language and Information

**APPROVED FOR PUBLIC RELEASE:
DISTRIBUTION UNLIMITED**

This paper was originally published in *Annual Review of Computer Science, Volume 1*, J. F. Traub (ed.), Palo Alto, CA: Annual Reviews Inc. and is reproduced here with the permission of the publisher. Its preparation was supported by the Defense Advanced Research Projects Agency under Contract N00039-84-K-0078 with the Naval Electronic Systems Command.

SRI International®

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
22 AUG 1986		**00-08-1986 to 00-08-1986**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Natural-Language Interfaces	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **SRI International,333 Ravenswood Avenue,Menlo Park,CA,94025**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**48**	

Table of Contents

Natural-Language Interfaces

C. Raymond Perrault and Barbara J. Grosz

Artificial Intelligence Center and
Center for the Study of Language and Information
SRI International
Menlo Park, CA 94025

1. Introduction

Since the early 1960s when support decreased for machine translation, much of the research on natural-language processing (NLP) in North America has been motivated by its potential use for communicating with software systems.[1] Natural-language systems have been developed to extract information from databases, to control (simulated) robots [104], to interact with graphic systems [8], to specify simulation problems [36], and to communicate with systems embodying expertise in some task or problem area [7, 83].

In this article we focus on interfaces to database management systems (DBMS).[2] We use the term **natural-language interface** (or NLI) to refer to such interfaces, unless otherwise specified. In addition to being among the earliest interface systems developed, interfaces to databases account for most of the NLIs implemented to date and they are the subject of a substantial literature. Although some work has been done on the use of natural language to update databases [25] and on generating appropriate responses, most of the work on NLIs has been concerned with interpreting queries, and we will restrict ourselves to this problem area.

Besides discussing the main system architectures used in NLIs, we also sketch the body of techniques developed for them. In doing so, we distinguish between the **task** of an interface (the various functions of the underlying software system, such as answering questions, updating a database, or moving a robot) and its **domain** (the set of objects, properties, and relations) denoted by the utterances it must interpret e.g. (employees and managers).

---

[1] Notable exceptions include the story understanding programs of Schank and his colleagues [13, 86]

[2] We do not discuss commercial systems even though they are becoming increasingly available [5, 51]; the first was ROBOT/INTELLECT [35].

Natural language (NL) is but one of the methods available for human-machine interaction, but the reasons for its attractiveness are obvious:

- It provides an immediate vocabulary for talking about the contents of the database.
- It provides a means of accessing information in the database independently of its structure and encodings.
- It shields the user from the formal access language of the underlying system.
- It is available with a minimum of training to both novice and occasional user.

Although form-filling and menu-based techniques [94] are appropriate to simple software systems whose structure is easily learned (and whose only user may be its designer), we conjecture that NL becomes more desirable as the following become true:

- The organization of the underlying information and procedures becomes more complex, so that the information necessary to process one query may be distributed widely throughout the system.
- The encoding of the information becomes more remote from everyday concepts, perhaps for the sake of retrieval efficiency.
- The problems the user wishes to solve become so complex that even *writing* a correct program in a formal query language may be difficult.

For example, the English query "Who owns the fastest submarine?" translates into over 20 lines of code [39] in the query language DATALANGUAGE. Even when compared to the more abstract relational query languages, NL is more concise. For instance, Warren and Pereira [99] provide the following QUEL [93] equivalent for the query "How many countries are there in each continent?"

```
range of C is countries
range of Cont is continents
range of I is inclusions
retrieve (Cont.name, count(C.name
                        where C.name=I.inside and
                              I.outside=Cont.name))
```

As indeed they must, NLIs allow the same information to be requested in a variety of ways. For example, the following queries might all be used to ask a database to determine which manufacturers were known to have shipped equipment to Mexico:

Who sent equipment to Mexico?
Who sent Mexico equipment?
Mexico received equipment from which manufacturers?
Equipment was sent to Mexico by whom?

The function of an NLI is to translate utterances in NL to expressions of a more immediately interpretable form, such as the formal query language (QL) of a DBMS. In this regard the NLI is much like a programming-language (PL) compiler although differing from it in some important respects. The syntax of a PL is much simpler and the language is intentionally free of both syntactic and semantic ambiguities. PLs and their compilers assume certain primitive data types (e.g., numbers, strings). Although programs written in these PLs may be about other types of objects (e.g., employees, salaries), the syntax, the semantics, and the compiler of the PL are not sensitive to these types; the programmer must explicitly provide an encoding for them into the data types provided by the PL. NLIs, on the other hand, are inherently sensitive to the types of objects in the domain. Thus, whereas with PLs the programmer must encode the objects in the datatypes of the PL, with NL the decoding burden is on the interface designer.

To simplify the discussion, we assume throughout that the underlying DBs are relational [15] and that the query language is relational calculus [16]. The relation between other DB models and the relational model is well understood [96]; at worst they can be accommodated by building translators to them from relational calculus.

In the following section, we introduce a small database as the basis for the examples in this paper and we examine some of the more important problems of interpretation that an NLI must be designed to handle. We discuss the main sources of information available for the interpretation of utterance and outline the general features of the architecture of three classes of NLIs. We then offer a more detailed description of various NLI constituents, which shows how the sources of information are used by different systems to solve the various problems of interpretation. We conclude with a brief review of current research issues in NLP and their importance for more sophisticated interfaces to software systems.

2. An Overview of the Problems

The flexibility and succinctness of NL for querying DBs are achieved at the cost of problems in determining the interpretation of a query.[3] Several of these problems, which we illustrate briefly here, have received interesting general treatments within the context of NLIs. For purposes of illustration, we consider a simple database containing information about employees and divisions in an organization. The information about

[3]Succinctness is certainly not a characteristic of all uses of NLP: for example, it is not a property of NL when used for the direct specification of low-level programs.

an employee includes name, salary, division, and whether or not the employee was exempt from overtime pay. The information concerning a division includes its manager, its revenue, and its product.

The syntactic structure of a sentence is often ambiguous. For example, in the request, "Give me all the employees in a division making more than $50,000," it is unclear whether the modifying phrase "making more than $50,000" is meant to apply to employees or divisions. This may be termed the **modifier attachment problem.** In some cases, however, certain possibilities can be filtered out on semantic grounds. For example, while in general, "making shoes" in the query "Give me all the employees in a division making shoes" could modify either "employees" or "division," in a domain constrained by the information our the sample database, only divisions make shoes, not employees; thus the query in this specific case is unambiguous.

NL sentences with determiners--words such as "the," "each," and "what" can have several readings, unlike the well-formed formulas of quantified logic. For example the query "What employee earns more than every division manager?" might be either a request to name the one employee whose salary exceeds that of any division managers or a request to name for each manager some employee who earns more than that manager. The relative scoping of the quantifiers corresponding to the different determiners depends on a number of factors, including the form of the utterance, the particular determiner, and the context of use. Various solutions to this problem, which is referred to as the **quantifier scoping problem**, are presented below.

The **nominal compound** problem is illustrated by the phrase "sales division" in the query "Who manages the sales division?" Such noun-noun combinations occur frequently in natural language. The syntax itself gives no clue as to the relationship between "sales" and "division." This kind of construction can be used to express arbitrary relationships (as illustrated by combinations like "wine glass," "oil pump," and "pump oil") and can be extended to longer concatenations of nouns ("national park ranger station equipment procurement form"). The syntax does not even determine the direction of the modifier relationship (editors' attempts to encourage helpful hyphenation notwithstanding). For example, "Stanford Research Institute" formerly referred to a research institute associated with Stanford University, whereas "Computer Research Institute" would likely refer to an institute organized to conduct computer research. This problem is one of several related to modification discussed below.

The interpretation of a query may depend in a number of different ways on previous queries and their interpretations. Of these forms of dependency, elliptical utterances and certain uses of pronouns are prevalent in database querying.

Elliptical queries often arise because users are interested in obtaining similar information about different objects. After making a full request, they may ask for

additional information with a single word or phrase. For example, Query (1) below can be followed by either of the elliptical queries, (2a) or (2b), which should then be interpreted as (3a) or (3b), respectively.

1. Who is the manager of the automobile division?

2a. of aircraft?
2b. the secretary?

3a. Who is the manager of the aircraft division?
3b. Who is the secretary of the automobile division?

In these two examples, the "expanded" query is like the original one with but a single word (different word in each case) replaced. The kind of expansion required may be much more complex, however; for example, a simple constituent may have to be replaced with a more complex one, as in Queries (4) and (5) below; or different parts of the original query may require replacement as in Queries (6) and (7).

4. What is Benson's salary?

5a. the sales division manager's
5b. the highest revenue division's manager's?

6. What are the salary and title of the highest paid nonexempt employee?

7. Division of the lowest paid?

Note that query (7) might be interpreted as either (8a) or (8b).

8a. What are the salary and division of the lowest paid nonexempt employee?

8b. What is the division of the lowest paid nonexempt employee?

Pronouns and other referring expressions provide one means of referring repeatedly to the same entities. For example, "they" in Query (9b) must be **resolved** to refer to employees who earn more than the sales division manager.

9a. Can you tell me which employees earn more than the sales division manager?

9b. How much do they earn?

3. Constraints on Interpretation

In computational linguistics, as well as linguistics more generally, there is substantial disagreement (and no small amount of confusion) as to what interpretation actually is. Agreement has yet to be reached on answers two fundamental questions:

- What receives interpretation? The alternatives include sentences, sentences in context, sequences of sentences, and dialogues.

- What is its object? Here alternatives include truth-values (especially for declarative sentences), answers (for questions), procedures for giving answers, or even the mental state the speaker must be in to make his utterance.

Within the restricted realm of interfaces to DBs, it is generally taken to be sentences and, occasionally, sequences of sentences that receive interpretations. The interpretation given to a query is taken to be a complex predicate; this predicate is satisfied by all the tuples of objects that are answers to the question. To allow for the possibility of ambiguity, we will take interpretation to be a relation between sentences and these complex predicates. For the interpretation relation to be specified, the following must be provided:

- A number of **information sources,**[4] each consisting of a class of **objects** and **constraints** on those objects. Thus, the syntactic information source might have words, phrases, and features as objects, and syntactic rules as constraints.

- **Constraints** that hold across information sources--expressing, for example, the relation between parse trees and their associated senses, or between sets of words (from the morphology) and sentences (from the syntax).

The NLI designer must also decide how the various objects and constraints will be represented, and how interpretations or, more accurately, their representations will be computed. One confusion that abounds in much of the computational-linguistics literature is the identification of interpretations with representations (i.e. interpretations are taken to *be* representations).

Although it is desirable for the overall theoretical account to be as modular as possible, computational efficiency may (and often does) suggest architectures where the various sources of information interact significantly. The kinds of information that are considered depend upon the kinds of tasks being performed by the NLI and the linguistic proficiency that is being sought. The standard information sources include

---

[4]These are often called *knowledge sources*, but we prefer to reserve the term *knowledge* for other uses, as it suggests that the information is true; this is a connotation we wish to avoid.

morphology, syntax, the lexicon, illocutionary and discourse information, and encyclopedic information about the domain.

The objects of **morphology** are words, their roots, inflections, and derivations. Inflections in English include markers for number (to distinguish the singular "employee" from the plural "employees"), gender (to distinguish the masculine "him" from the feminine "her"), and case (to distinguish the nominative "who" from the accusative "whom"). Derivational morphology accounts for relationships among words of different syntactic classes, such as "inflate," "inflation," "inflationary," and "disinflate." Many NLIs include some treatment of inflectional morphology to minimize the size of the lexicon. Winograd [105] provides a simple procedure. A more sophisticated computational treatment based on finite-state transducers is presented by Koskenniemi [59].

The objects of **syntax** are words, phrases, and features. Of particular concern are phrase types (to distinguish noun phrases, prepositional phrases, and verb phrases), constraints on phrase structure (for example, that a prepositional phrase such as "in the auto division" consists of the preposition "in" and the noun phrase "the auto division"), and various phenomena collectively labelled as long-distance dependencies. These include constraints on complements (such as that John is the person doing the pleasing in "John is eager to please" but is the one who is pleased in "John is easy to please"). We include a brief review of various syntactic issues below; Winograd [105] provides an excellent detailed treatment.

The **illocutionary** source is concerned with the actions (e.g. assertions, questions, requests) that can be performed by using language, and with the indicators of those actions. In written language, the principal indicator is sentence mood whether a sentence is indicative, interrogative or imperative. In spoken language, intonation is also important.

The **discourse** source specifies how the context established by sequences of utterances interacts with interpretation. It includes constraints on the structure of the sequence that are provided by linguistic expressions, as well as constraints on the interpretation of particular phrases that derive from the form and content of previous utterances.

The **encyclopedia**[5] contains constraints derived from the "real world"; it specifies its objects, relations, the structure of events, and the content of mental states. Of particular importance to NLIs is the **domain model**, that part of the encyclopedia describing the domain of the DB. The encyclopedia also encodes restrictions on what

[5]This is often called *real-world* or *commonsense knowledge*.

word senses can modify or be modified by what others (e.g. that the adjective "solvent" can apply when to "bank" denotes a financial institution but not when it denotes the side of a water course), and sortal restrictions indicating that in "John paid Mary" the syntactic object "Mary" is the recipient of the payment, while in "John paid 5 dollars" the syntactic object "5 dollars" is the amount of the payment.

NLIs, unlike general linguistic theories, also need information about the software system to which they are interfaced. We simply call this **database** information.

Constraints are also necessary to relate information across information sources. The first set of these is the **lexicon**, which specifies relations between words and their senses (e.g. that the word "bank" has at least the two senses mentioned above). Also important are those constraints stating how to derive the interpretations of various syntactic constructions from those of their constituents. In some cases, these constraints relate parse (sub)trees with interpretations, while in others syntactic and semantic rules are linked.

Solutions to the interpretation problems mentioned in the previous section must typically make use of several information sources. The referent of a pronoun, for example, is constrained by syntactic, lexical, encyclopedic, and discourse information.

We have so far avoided the term *semantics*. In accordance with common practice in the field, we will use *semantics* in three ways, generally leaving it to the context to distinguish uses. By the **model-theoretic semantics** of an utterance we mean its interpretation, subject to the constraints of the information sources. We also refer loosely to the lexicon, encyclopedia, and illocutionary sources as **semantic sources**, or simply **semantics**. Finally, the process of finding a representation for what we call here the interpretation of an utterance is generally called **semantic interpretation**.

Most of the current attempts to develop a model-theoretic semantics for NL, roughly parallel to that given to artificial languages, are inspired by the work of Montague [69]. Although Montague's interpretations could at least in principle, be assigned directly to sentences, his formulation did make use of an unambiguous intermediate formal language--the language of intensional logic. In the computational framework such intermediate languages, or **logical forms**, are common. Moore [70] examines various problematic NL constructions (e.g., adverbs, tense, quantification, and questions) and suggests ways of encoding them in a higher-order predicate calculus with intensional operators. Encoding of information in sematnic sources lies at the very heart of artificial intelligence research. The articles in Hobbs & Moore [48] discuss a number of such encoding problems, from the perspective of first-order logic and its extensions.

The use of logical languages for representation and of formal deduction as the

means to draw inferences, as well as the desirability of a model-theoretic semantics for NL (and for the representations constructed in the process of interpreting utterances) are still controversial. Most studies in NL processing until the late 1970s, and many current efforts as well, stress the computational aspects of determining an interpretation rather than semantic issues [86, 103, 44, 74]. Much of this research emphasizes the role of implicatures based on stereotypical and salient information.

4. System Architectures

The various architectures in NLI systems reflect different choices of what information is to be applied (and thus what interpretation problems to attempt) and in what manner. After sketching the three main architectures, we discuss their differences and how these affect the range of natural language they can handle.

All systems must build at least one internal representation of a query, that is, an expression in QL. Some systems add an explicit, purely syntactic representation: one of the earliest and best known of these is Woods's LUNAR [109], described briefly in the following section. Semantic grammar systems, further discussed in the next section, also produce only a single intermediate representation, which in this case encodes constraints from several information sources. Finally, many systems produce a separate representation of the meaning of the query in terms of the concepts of the domain of the DB, independently of the DB structure. We use the term **intermediate-representation language** (or IRL) to refer in general to the languages in which these representations are expressed; the particular names of IRLs in individual systems (e.g., meaning representation language, logical form) are used only when discussing the particular properties of those systems.[6]

4.1. LUNAR

The LUNAR system [109], based on earlier work by Woods [107], pioneered many of the techniques that still underlie most NLIs. Designed as an interface to a two-file database containing information about chemical analyses of the Apollo-11 moon rocks and references to the literature on those analyses, LUNAR has three components: a parser, a semantic interpretation routine, and a query interpreter. The parser uses an augmented transition network grammar (discussed in more detail in the section on syntax) to produce parse trees in the form suggested by Chomsky [14]. The grammar is a domain-independent grammar of English, which, through subsequent development as part of several systems, has become one of the most extensive computer-based English grammars ever constructed.

---

[6]The optimization of the generated queries is not discussed in this paper. Whether or not IRLs are used does not affect the question of whether, but only when and in what manner optimization can be done.

Semantic interpretation rules are used to map parse trees to QL expressions. Generally triggered by the head of a constituent (verbs for sentences, nouns for noun phrases), the rules obtain interpretations of the dependent and modifying constituents; they then combine these into the interpretation of the whole. Thus, there will be a set of semantic interpretation rules for each noun and verb in the sublanguage covered by the NLI.

The target of the semantic interpretation is an expression in a modal first-order quantified language; this expression can be evaluated directly against the database to return a set of records. The vocabulary of the QL includes all the relations encoded directly in the DB, plus a number of derived relations. The only constraint on derived relations is that it should be possible to associate with each of them its own retrieval function, expressed in terms of the basic relations of the DB.

Figure 1 shows both the parse tree and the resulting QL query produced by LUNAR for the sentence "Which rocks contain chromite and ulvospinel?" LUNAR's parses are not surface structures; in this query, the question-determiner noun phrase "which rocks" is taken to be the logical subject of the sentence and the analysis is analogous to that of "which rocks such that they contain chromite and ulvospinel exist?" The QL query includes two database-query specific constructs: SEQ, a general purpose enumeration function that assumes its argument is a (precomputed) list, and PRINTOUT.

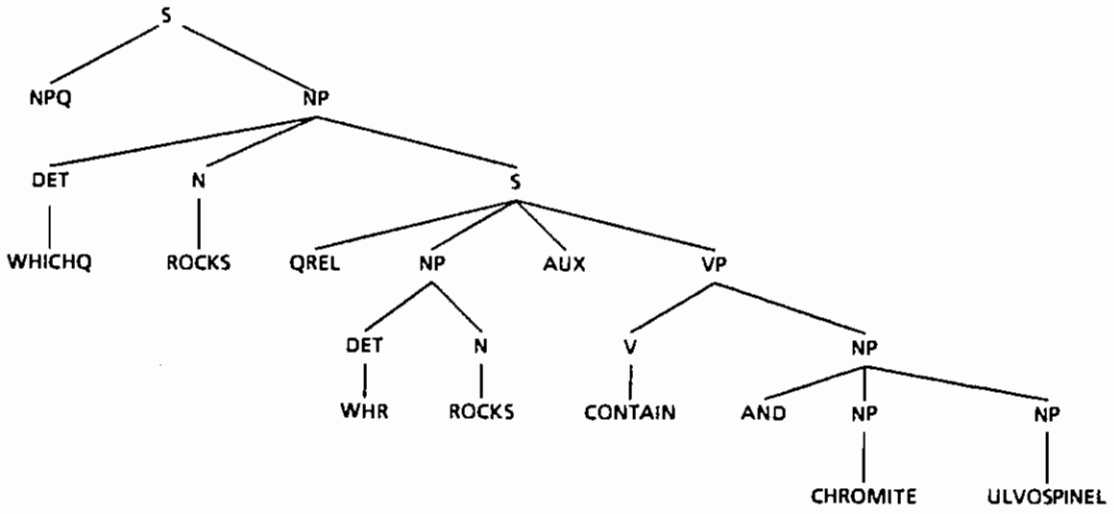
After LUNAR, architectures of natural language processors NLPs diverged in two directions: systems were constructed in which either (a) syntactic, lexical, encyclopedic, and database information was encoded in one set of rules, or (b) different information sources were kept quite separate. We examine each of these in turn.

4.2. Semantic-Grammar-Based Systems

The principal characteristic of a **semantic grammar** [11] is that it intentionally collapses distinctions among information sources. NLIs that incorporate semantic grammars vary somewhat in the details, but all classify words and phrases under a combination of syntactic, lexical, illocutionary, and database information. Exemplars of different approaches are PLANES [98], LADDER [39] and REL [95]. The grammar rules incorporate categories that are oriented around a particular domain and task.[7] For example, a semantic grammar for the domain of university life might contain the categories *student*, *instructor*, and *course-times*; one for the domain of ships could include *ships*, *officers*, and *ship-locations*. In contrast, typical categories of syntactic grammars are "sentence" and "noun phrase". A semantic grammar for the task of

---

[7] As there is nothing especially *semantic* about these grammars, the term "aggregate grammar" might be less confusing.




```
(FOR EVERY X7 (SEQ VOLCANICS)
                    (AND (CONTAIN X7 (NPR* X9 'SPINEL))
                         (CONTAIN X7 (NPR* X10 'CHROMITE)));
                    (PRINTOUT X7))
```

Figure 1: Parse tree and QL query from LUNAR

database querying would have a category to cover the presentation of answers; this category might include various interrogatives (e.g. "what is") as well as certain imperatives (e.g. "show me"). In contrast, a semantic grammar for an experimental setting might include a category that covered references to hypothetical situations (e.g. "if ...," "what if...," "suppose that..."). Associated with each "syntactic" rule in the semantic grammar is a rule for combining the results of the interpretations of the subconstituents into an interpretation of the consitutent being analyzed.

As an example, we can consider a simple semantic grammar for handling queries about our sample database. To handle the query "Who manages the automobile division?" the grammar would include rules like the following:[8]

Grammar Fragment

<SENTENCE> → <PRESENT> <ATTRIBUTE> <DIVISION>
(db(subst (genvar '' 'DIVISION ATTRIBUTE')))*

<PRESENT> → who (is) / what (are) / show (me)

<ATTRIBUTE> → <ATTRNAME>

[8]The grammar rules and lexical categories are in roman type, the associated interpretation in italics.

'return ATTRNAME.'*

<DIVISION> → the <DIVNAME> division
*'for each * in DIV file with DIV-NAME.*= 'DIVNAME''*

Lexicon Fragment

manages: <ATTRNAME>
'manager'

automobile: <DIVNAME>
'auto'

Figure 2 shows the "syntactic analysis" and the interpretation for the above query. Each node of the tree is associated with an interpretation for the subtree below it; for example, the node labelled <ATTRNAME> would (from the lexical information) get the interpretation, 'division' and the node <ATTRIBUTE> would (from the third rule) get the interpretation 'RETURN MANAGER.X'.

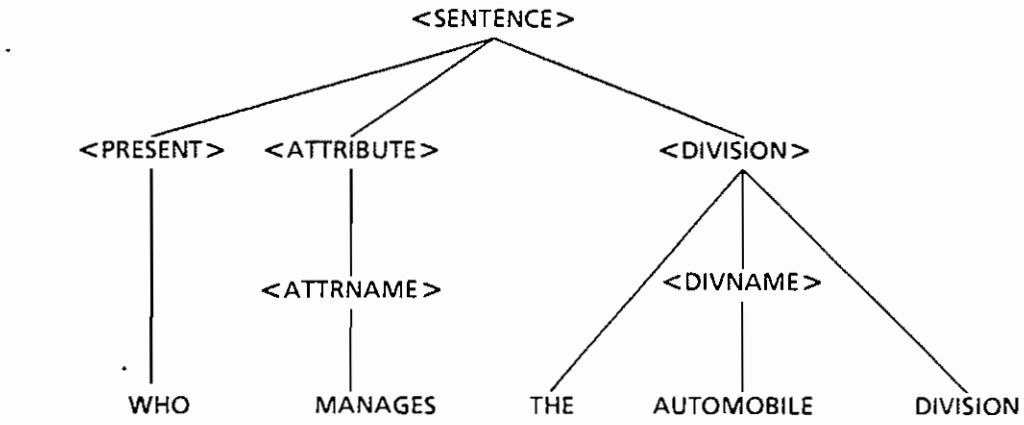


Figure 2: Parse tree and QL query from a semantic grammar

Unlike the nodes in the parse tree produced by LUNAR, the nodes in this parse tree are not labelled with general syntactic categories. However, as in LUNAR (and to

an even greater extent in some cases), the interpretation here assigned to a query is essentially a piece of code that states how to retrieve the answer to the query.

As is evident from this example, a semantic grammar is both domain- and task-dependent; a different grammar must be constructed for each application. The LIFER system [38], on which LADDER was built, supplies a set of tools for building semantic-grammar-based NLIs. Although LIFER provides general capabilities for handling ellipsis and paraphrase (the first is done by the parser and hence works for all LIFER-defined grammars; paraphrases are handled by automatically modifying the language definition), it too requires a new grammar for each different application domain and task.

4.3. IRL systems

IRL systems (CHAT-80 [99], IRUS [5], PHLIQA1 [85, 61], TEAM [34] and Ginsparg's [31]) construct at least three separate representations of a query: a parse tree, an IRL formula, and a QL query.[9] Each system separates the rules stating syntactic constraints from those that specify lexical, semantic, encyclopedic, and discourse constraints. Typically the objects, predicates, and relations of the encyclopedia furnish the IRL's basic vocabulary, and the representations used for encyclopedic constraints are quite close to those used for the QL. Encyclopedic constraints include at least taxonomic information (types and subtypes) and constraints on the arguments of predicates and relations.

The differences between the IRL- and other architectures can be clarified by an example. For the query "Which countries contain a volcano and a nonvolcanic peak?" an IRL system[10] would produce a parse tree like the one in Figure 3 by using such grammar rules as the following:

```
SWHQ → WHNP PREDICATE
VP → VPT  NP
NP → DETP NOMHEAD
NP → NPSERIES CONJ NP
```

The parse, like LUNAR's, is based on a general grammar of English. (However, it

---

[9] From this perspective, the PLANES system is a hybrid; it uses a semantic grammar but actually builds an intermediate representation of the "meaning" of the query from which it constructs the QL query. Because its IRL, like its grammar, is designed specifically for the task it undertakes (i.e., it comprises a collection of special-purpose "frames"), we have included it with the other semantic-grammar systems.

[10] We will use an example produced by the TEAM system; the actual structures produced by other IRL systems would, of course, differ in detail.

is a surface structure, not a deep structure, analysis, reflecting a change in underlying syntactic theory.) For example, the conjunction "a volcano and a nonvolcanic peak" is treated as a conjunction of noun phrases as was the conjunction "chromite and ulvospinel" in the LUNAR example.[11]

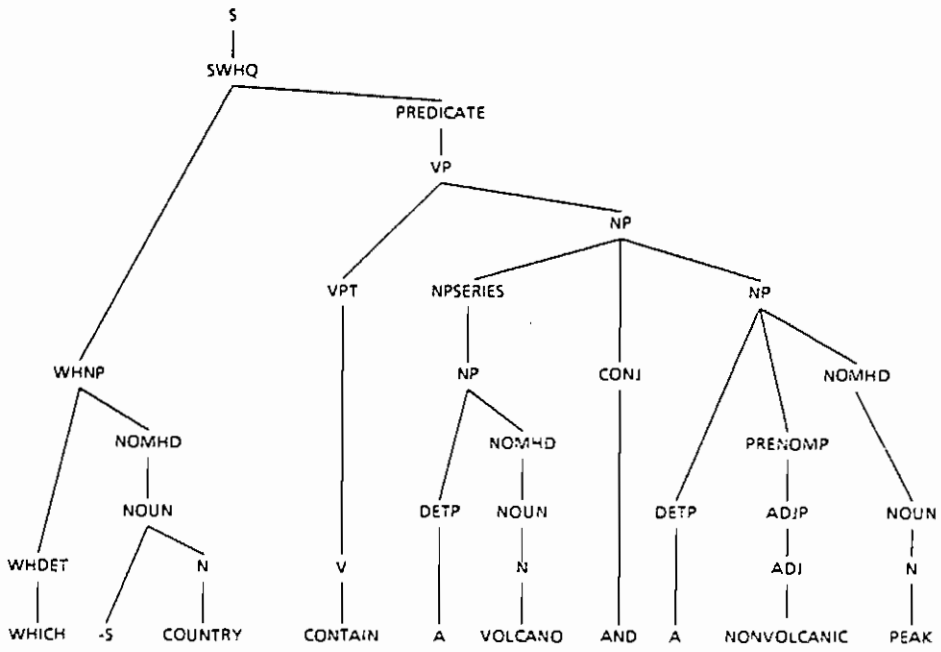


Figure 3: Parse tree from IRL system

The IRL representation of the interpretation of the query (in this case logical form) is shown in Figure 4 along with the QL (in this case an expression in SODA [72]). The IRL representation is a complex predicate composed of general predicates in the domain; it makes no reference to the actual database structures or any retrieval process. Only the QL representation reflects the database and the querying task. Although there are fragments of the LUNAR QL that resemble the logical form (e.g., the representation of the meaning of the conjoined NPs), the overall representations are different in kind.

4.4. Comparing Architectures

The different architectures provide for different ways of handling various interpretation problems. We leave until the next section discussion of the particular ways they do so. There are five major overall differences among the architectures.

---

[11] In semantic-grammar-based systems, conjunction, if treated at all, is specialized for aggregate categories, containing rules such as <DIVISION> → <DIVISION> and <DIVISION>.

```
(QUERY (WH COUNTRY1
          (COUNTRY COUNTRY1)
          (SOME PEAK-VOL3
                   (PEAK-VOL PEAK-VOL3)
                   (SOME PEAK4
                            (AND (PEAK PEAK4)
                                    (NONVOLCANIC PEAK4))
                            (AND (CONTAIN COUNTRY1 PEAK-VOL3)
                                    (CONTAIN COUNTRY1 PEAK4))))))


((IN #:$1 PEAK)
       ((#:$1 PEAK-VOL) EQ Y)
       (IN #:$2 PEAK)
              ((#:$2 PEAK-COUNTRY) EQ (#:$1 PEAK-COUNTRY))
              ((#:$2 PEAK-VOL) EQ N)
              (? (#:$1 PEAK-NAME))
              (? (#:$2 PEAK-NAME))
              (? (#:$1 PEAK-COUNTRY)))
```

Figure 4: IRL and QL representations from IRL system

First, the information sources that contribute to the interpretation of a query by the system are different. Many systems, for example, make little (or only ad hoc) use of morphological, illocutionary, or discourse constraints. In one way or another, however, they all utilize syntactic, lexical, and database constraints.

Second, there are different ways of combining the information sources into sets of rules. The semantic-grammar systems combine all sources into one set of rules. LUNAR distinguishes syntactic rules from the rest. IRL systems also separate database information and provide general constraints for mapping between syntactic constructions and their interpretations.

Third, the application of separate sets of rules may be sequential or interleaved. Although most systems apply the rules sequentially, IRUS uses the capabilities of the RUS parser [6] to interleave syntactic and semantic constraints; the interleaving is accomplished with cascaded ATNs [111]. Interleaving is done in Colmerauer's Prolog-based system [20] and was also used in several speech understanding systems [63, 97].

Fourth, the range of queries the systems can process at different stages is different. In semantic-grammar-based systems, any query that can be parsed can be translated into QL. In contrast, LUNAR and IRL-systems can syntactically analyze some sentences for which they cannot construct a semantic interpretation. The range of

concepts covered also differs. In semantic-grammar-based systems, only those queries that can be translated in QL can be interpreted at all. In contrast, in IRL systems, the concepts (i.e., objects, properties, relations) in the domain model provide the basic vocabulary for the IRL. A mapping from these concepts to DB structures provides the basis for translating IRL expressions (which are in terms of the concepts of the domain model) into QL expressions. With this sort of approach it is possible to supply interpretations of queries for which there is no QL representation (e.g., because the DB covers the domain only partially.)

The IRL systems all take this type of approach; the actual coverage they offer, however, depends on how their domain models are defined. For example, the PHLIQA1, IRUS and CHAT-80 domain models are provided completely independently of the DB (they are essentially "hand-built" by the system designers); it is therefore quite possible for them to construct IRL representations of queries for which there is no QL representation. In contrast, the TEAM system, which automates the process of adapting an NLI to a new domain and DB, constructs its domain model mechanically from information supplied about the DB; this restricts the concepts to those that can be generated from the DB concepts through relational calculus.

Finally, the architectures differ with respect to how easy it is to adapt an interface to a new domain or DB. As remarked previously, a semantic-grammar-based system requires extensive revision to be adapted to a new domain or task. Because all constraints are encoded in the grammar, the grammar itself must be rewritten or at least extensively revised. In contrast, adapting an IRL system to a new database requires little, if any, change in the syntax rules. In some systems (IRUS, Ginsparg's, PHLIQA1) modification of the semantic rules is required; in others (TEAM, CHAT-80) the semantic rules do not change; only the domain model and lexicon do.

5. Methods

A number of techniques have been developed for encoding and applying the information sources needed to determine the interpretation of a query. In this section, we examine various methods used to handle the interpretation problems discussed earlier. We have chosen to focus on techniques sufficiently general for a wide range of natural-language-processing applications. As a result, certain problem areas are covered in more detail than others. This unequal treatment reflects, in part, a difference in the state of the art in the various areas of NLP. The usefulness of any specific method depends to some degree on a system's architecture; where it is relevant and not obvious, we will remark on the applicability of a method to different architectures.

5.1. Syntactic Models

With very few exceptions, phrase-structure grammars have provided the basis for the syntactic components of NLIs. Most of these grammars, in fact, are context-free (CF), with the possible addition of extra conditions on the subconstituents. The languages generated even by the extended grammars are, almost certainly, CF. In fact, the only solid arguments contending that NLs are not weakly CF are quite recent (Shieber [90] for Swiss-German and Culy [22] for Banbara.) Both involve constructions not treated by grammars in existing NLIs. As with programming languages, non-CF grammars may be used to make the description of CF languages easier, especially when some constraints (subject-verb agreement, subject and object control) must be applied to non-adjacent nodes in the parse tree. Perrault [80] surveys the known formal properties of some of the more common syntactic formalisms. Slocum [92] compares the performance (on several hundred sentences) of various parsing strategies.

The first substantial extension of CF grammars widely used in NLP was the **augmented transition network grammar** (ATNG) of Woods [108]. The ATNG is a two-step generalization of the Finite-State Automaton (FSA) [49]. The FSA has a finite set of states; transitions among them are allowed when certain symbols appear in the input. One of the states is distinguished as the start state, one or more as final states. The input string is accepted if it leads to a sequence of acceptable transitions from the start state to a final state. The languages recognized by FSAs are the finite-state, or Type 3 languages. **Recursive transition networks** (RTN) generalize FSAs by allowing a transition between two states to be taken via a recursive jump to a start state. RTNs recognize exactly the class of CF languages. Finally, the ATNG adds to the RTN a finite set of **registers** and **actions** that can set registers to words observed in the input, their corresponding lexical entries, or to some function of the contents of other registers; a recursive call to the network can pass values back to its calling level, which can in turn assign that value to a register. Transitions can be made conditional on register contents. ATNGs generate all recursively enumerable sets.

Because grammars for all but the smallest subsets of NLs are ambiguous, the LR(k) techniques often used for parsing PLs are generally not applicable to NLs. In their place, a number of parsing algorithms have been developed.

ATNGs are naturally implemented in recursive top-down parsers; in fact, in the early literature on the subject, grammars and parsers were hardly distinguishable from one another. The register assignment mechanism makes it difficult to conceive of using the grammar in other than a top-down left-to-right parsing scheme.

Much effort was devoted to efficient implementation of top-down ATN parsers. In the early implementations, the grammar and the lexicon were encoded as LISP data structures and interpreted by the parser. Burton & Woods [12] then showed how to compile the parser and the grammar into a large LISP program, and then, through the

LISP compiler, into machine language. Compilation improved parsing performance by an order of magnitude.

However, pure top-down parsers suffer from some well-known problems. First, they cannot handle left-recursive constructions (as in "John's father's brother's book"), and second, their backtracking regimes may be very inefficient. The left-recursion problem can be solved by converting the grammar to a weakly-equivalent right-recursive one, but at the cost of complicating the process of deriving the interpretation.

The backtracking problem has been addressed in two quite different ways. The first has been through extensions of bottom-up (Cocke-Kasami-Younger [112]) and Earley [26] parsing strategies to non-CF grammars. These methods include use of the well-formed substring table [60, 106] and charts [55].

The second, and more radical, line is based on Marcus's Determinism Hypothesis. Marcus (1980) [65] that English (and possibly other NLs) can be parsed by a mechanism that operates "strictly deterministically," in that

- All syntactic structures created by the parser operating on an input string are permanent and must be included in the output produced for that input.

- The internal state of the mechanism is constrained so that it cannot encode temporary syntactic structures.

Marcus designed a parser satisfying these conditions (along with a small grammar for it) that captures interesting generalizations related to such phenomena as passives, imperatives, and yes/no questions. He also suggests a simple explanation for so-called garden path sentences, such as "The horse raced past the barn fell" and "Have the students who failed the exam take the supplemental" (closely related to "Have the students who failed the exam taken the supplemental?"). These sentences are perfectly grammatical, but their analysis by humans seems to require conscious backtracking. The determinism hypothesis is not without problems. For example, it depends essentially on an integration of syntactic and semantic analysis that remains to be demonstrated convincingly; moreover, no large deterministic grammar has yet been written. However, Marcus's work has influenced the design of some ATN parsers that now utilize look-ahead to reduce backtracking [6]. Recently, he suggests representing syntactic analyses as logical formulas over the domain of syntactic nodes, in which the disjunction of the possible attachments can be stated, or in which no attachments are stated at all, save those that preserve the left-to-right order of constituents in the sentence [66].

Another problem with ATNs was that the dependence of the *grammar* on left-to-right processing made it very difficult to use the same grammar with different control regimes. For example, if subject-verb agreement was to be tested by having the parser

assign to an ATN register the number of the subject noun phase, so that this register could then be tested upon encountering the main verb, this procedure would fail if the parser encountered the verb before the subject. In doing research on speech-understanding systems, Paxton [75] and Wolf & Woods [106] investigated parsing "middle-out," that is, starting from the highly stressed parts of the sentence and constructed parsers that were not order-dependent. In a different vein, some workers on language generation [54, 2] have argued that it is desirable to be able to make decisions about syntactic constituents independently of the order in which they are to appear in the utterance. It is not possible to do this, however, with an order-dependent ATN.

Although the need for order independence is still controversial (see Wolf & Woods [106] for speech recognition and McDonald [68] for language generation), several proposals to achieve it have been made, relying on **unification** of graphs as the main operation in parsing. One of the earliest proposals in this direction was Kay's functional-unification grammar (FUG) [56]. In several of these formalisms, grammatical rules are represented as formulas in first-order logic, or more accurately, in its Horn clause subset. In these logic grammars (under various guises known as metamorphosis grammars [19], definite-clause grammars [78], extraposition grammars [76], modular grammars [67] and others), predicates are defined to be true of strings meeting certain conditions, such as NPs. Nonlocal syntactic constraints and semantic constraints can be imposed by enabling the predicates to take on extra arguments allowing information to be propagated across the analysis. Subject-verb agreement provides a very simple example. Consider the following very simple grammar, expressed as first-order sentences. According to the conventions of Prolog, identifiers starting with an upper-case letter are variables and all free variables are assumed to be universally quantified. The indices I, J, and K take integer values denoting positions between words in a sentence.

```
s(I,J,Number) ⇐ np(I,K,Number) & vp(K+1,J,Number)
vp(I,K,Number) ⇐  v(I,K,Number)
np(I,K,Number) ⇐  occurs(I,I+1,the) & n(I+1,K,Number)
n(I,I+1,Number)  ⇐ occurs(I,I+1,X) & lex(X,n,Number)
v(I,I+1,Number)  ⇐ occurs(I,I+1,X) & lex(X,v,Number)
```

If the lexicon contains the assertions

```
lex(fish, n, singular)
lex(fish, n, plural)
lex(fish, v, singular)
lex(swim, v, plural)
lex(swims, v, singular)
```

then the sentence "the fish swims" can be recognized as generated by the

grammar by asserting

```
occurs(1,2,the) & occurs(2,3,fish) & occurs(3,4, swims)
```

and then proving that

```
(∃ N) s(1,4,N).
```

The heart of logic grammars is their use of unification as a way to test the compatibility of information and to propagate constraints. Although definite-clause grammars, for example, provide all the necessary expressive power within Prolog, this power is achieved at the cost of a certain lack of perspicuity: the constraining predicates end up having as many arguments as there are "pieces of information" that they control or that must be propagated through them. These arguments are all specified positionally; in the example above, the first two arguments denote the delimiting positions in the input string, while the third denotes the number feature of the subject and verb. This can easily lead to very long argument lists whose management is difficult.

In the last few years, several more perspicuous unification-based syntactic formalisms have been developed that derive their inspiration from both the linguistic and computational traditions. From pure linguistics have come lexical-functional grammar [52] and generalized phrase-structure grammar [30], which are full syntactic theories, including formalisms for representing rules and derivations and general constraints on the use of these formalisms. Coming from the computational perspective, the already mentioned FUG of Kay and PATR-II [89] are formalisms only, without theoretical commitment.[12] The semantics of the formalisms has been studied with the tools of denotational semantics [87] by Pereira and Shieber [79]. Kay has investigated the use of FUG for both generation and recognition.

Writing the extensive grammars needed by useful NLIs is still a difficult task that is normally performed only in research centers with substantial resources. Some examples are the LUNAR grammar, revised through several projects at Bolt, Beranek and Newman and now part of the IRUS system [5], the DIAGRAM grammar [82], first developed at SRI as part of the SRI Speech-Understanding Project [97] and now included in the TEAM system, and the grammar of the Linguistic String Project [84].

Most "practical" grammar-writing exercises result in very liberal grammars that will accept sentences native speakers would not consider grammatical. There are three reasons for this. First, since grammars are devices that permit (rather than proscribe) membership in a language, it is often easier to write a small number of very general rules than a large number of specific ones. Second, it may be easier to exclude

---

[12] This is also the case with ATNGs and definite-clause grammars.

uninterpretable sentences on nonsyntactic grounds. Finally, one might want to allow certain nonstandard sentences (e.g., telegraphic speech) to be treated as if they were grammatical [102], if there is reason to believe that users would want to express themselves that way. The main practical drawback in such a liberal position is that, by proliferating parses, it becomes much more difficult to select one that is semantically acceptable.

No discussion of syntactic models would be complete without mention of the transformational grammars (TG) introduced by Chomsky [14]. They have provided the framework for much of the theoretical work on syntax since the 1960s. A TG has two main constituents: a base grammar, usually a phrase-structure grammar, and a set of transformations. The base grammar generates a class of trees, to which the transformations are applied to rearrange, copy, and delete constituents. The sentences of the language are the yield strings of the trees resulting from all possible applications of the transformations to all possible base trees. The details of the number and power of the transformations have changed considerably since their introduction in 1957, but, in some early versions of the theory, a passive sentence and its corresponding active sentence were transformationally related.

It therefore seemed plausible that one could build a parser that would take a sentence, construct a surface structure, and apply to it the transformations in reverse to obtain a base tree representing the interpretation of the sentence. This technique was first tried in a system built at MITRE [113] and then in the REQUEST and TQA systems built by Petrick, Plath and Damerau at IBM [24, 81]. One of the problems with the approach is that the inverse transformations can be applied only to the surface trees, even though the TG does not, in general, characterize those trees in any computable manner. The aforementioned systems dealt with this problem by handcrafting surface grammars. The TQA system is exceptional in that it is one of the very few to have been put to substantial use by bona fide users while it was undergoing development.

5.2. Semantic Interpretation

We turn now to semantic interpretation, the process of translating syntactic analyses into IRL.[13] The translation involves establishing three kinds of correspondences:

- Between the words of an NL and expressions in the IRL.
- Between various constituents of an NL phrase (e.g., head, subject, object,

[13]Some systems, including those using semantic grammars and several built by Schank and his colleagues [86, 62], never construct an explicit representation of the syntactic analysis but go directly from NL to IRL.

modifier) and the constituents of the expressions to which they correspond in the IRL (e.g., argument of a predicate, value of a field).

- Between the scope of determiners and other operators of an NL expression and the scope of the quantifiers to which they correspond in the IRL.

5.2.1. Vocabulary Correspondences

The first issue in semantic interpretation is the correspondence between words of the language and concepts in IRL. Some common nouns in English--such as "man" in "John is a man" -- correspond to one-place predicates in IRL and others--such as "manager" in "John is the manager of the sales department" correspond to relations. Verbs correspond to predicates--as in "John sleeps" -- or to relations--as in "John manages the sales department." Some adjectives, such as "exempt" in our fragment, can be interpreted as one-place predicates, although this solution is generally inadequate: adjectives such as "tall" must be interpreted differently, so that "tall men" and "tall babies" do not refer to things that are independently tall and men, or tall and babies. "Former senators" and "alleged thieves" are certainly not senators and possibly not thieves. In systems in which the IRL is first-order logic, the presence of these adjectives may affect the interpretation of the nouns they modify; when this occurs, the lexical-assignment problem interacts with the modifier-attachment problem. In LUNAR, for example, "analyses" and "modal analyses" are translated by two unrelated predicates. Prepositions correspond in some instances to relations (as in "What employees are *in* the sales department?"), while in others they are markers of the case of arguments of other predicates (as in "Did Bill go *to* Boston?"). Their interpretation varies according to the situation of use; Herskovits [41] provides an excellent discussion of locative prepositions (e.g., "on," "near," "beside") as well as a theoretical framework for handling them.

5.2.2. Modification and Attachment

There are various ways in which the meanings of constituents of a phrase can combine to determine, at least to some extent, the meaning of the entire phrase. Two special kinds of problems arise in computing these combinations:

- The surface form may not determine a unique association among the elements in a phrase; this happens, for example, with the attachment of prepositional phrases.

- Even when the association of constituents is clear, it may not be obvious exactly how the meanings combine; this may occur with combinations of adjectives and nouns, or with two nouns.

Proposed solutions to the attachment problem fall into three classes:

- The syntactic component makes direct use of lexical and encyclopedic

constraints and produces only attachments that satisfy all of them simultaneously.

- The syntactic component produces structures corresponding to all possible attachments, which are then filtered by other constraints.

- The syntactic component proposes one attachment only, representing all the alternatives, and the semantic interpretation component is allowed to move the attached phrase so as to satisfy the other constraints as well.

Semantic grammar systems adopt the first approach. Some logic grammar systems [20, 23] do likewise; these keep the syntactic categories separate, but have a single set of rules that constructs syntactic and IRL representations simultaneously. The second approach has the simplest organization and is used in many large systems such as LUNAR and TEAM. The third is used by CHAT-80. The last two approaches use **case frames** [10, 28] to encode the relations between verbs, their syntactic cases, restrictions on the types of the fillers of the cases, the target language predicate, and the correspondence between the syntactic case fillers and the arguments of the target predicate. Woods [110] and Pereira [77] contain excellent discussions of these topics.

The selection of IRL predicates to correspond to NL words has a considerable effect on the resolution of attachment problems. For example, the verb "have" can be used to express a have-as-part relationship ("A car has an engine"), an ownership relationship ("Susie has a Porsche"), and a have-as-property relationship ("Jack has red hair"), among others. This variety is also found with prepositions ("John is in the sales department," "John is in Europe"), genitives ("Joe's finger," "Joe's mother," "Joe's house," "Joe's friend") and nominal compounds ("American ship," "American car," "American cooking").

Although different kinds of surface forms give rise to these semantic problems, their treatment is similar in two ways. First, the resolution of the indefiniteness requires a search for the most reasonable relationship that can hold between two concepts. In the case of nominal compounds and genitives, these are the immediate constituents of the phrase ("Joe" and "finger," "American" and "car"), whereas for verbs ("have" and "be") and prepositions (e.g., "employees *in* sales") the two concepts being related are structurally more distant from each other. Second, the larger context of the discourse may make possible interpretations that would not arise in isolation. For example, although the phrase "Boston flights" would not ordinarily be taken to refer to flights that are only passing through Boston, in the two- query sequence, "Which flights from London to St. Louis enter the U.S. through Boston or Philadelphia? What times do the Boston flights leave?" the phrase receives precisely this interpretation.

Syntactic constraints determine which pairs of concepts need to be related for all

of these constructs except nominal compounds that include more than two nouns, but they do not further constrain the particular relationship. Because the relationship that may hold between the two concepts may be arbitrarily complex, some proposals for handling noun-noun relations in general [46] depend on sophisticated inferential capabilities and a complex model of the domain. Several techniques have been developed for handling a narrow range of such expressions under the assumption that users will not create new constructions (e.g., using the phrase "toilet paper submarine" to refer to a recently mentioned submarine that needs a resupply of toilet paper). Isabelle [50] surveys the nominal compound problem. Finin [29] presents a set of rules for handling those nominal compounds that can be resolved in terms of case relationships or type hierarchies. The TEAM system includes a limited treatment for nominal compounds as well as several other related problems that uses relationships derived straightforwardly from the database structure.

5.2.3. Scoping

The third set of interpretation questions involves determination of the relative scope in the target language of quantifiers corresponding to such NL determiners as "a," "the," "each," and "most" as well as to such operators as negation, tense, modals, and superlatives ("most," "oldest"). Viewed syntactically, the determiners occur in noun phrases, within the scope of verbs, but in first-order representations the quantifiers must be given wider scope than the predicates. Syntactically again, determiners can occur within one another's scope, as in "each manager of some division," or in parallel, as in "each manager manages some division." Operators can occur at the noun phrase level, such as in superlatives and in the negation in "none," or at the sentence level, such as in tense, modals, and sentential negation.

Even within noun phrases there may be changes in relative scope between the syntactic representation and the IRL: the interpretation of "Some employee of each manager is exempt" is that, for each manager, some employee of that manager is exempt. However, there are syntactic limits to how far up a quantifier can migrate: for example, no quantifier can move out of a relative clause, so that "Who is the manager who manages every employee?" cannot mean "For each employee, who is his manager?"

Aside from such syntactic constraints, all other relative scopings of the quantifiers are possible in certain circumstances, although some heuristics are useful for ranking the plausibility of the interpretations. Two can be mentioned. One simply gives preference to relative scopings, while preserving the left-to-right order of the corresponding determiners in the sentence. Thus, "Every manager manages some employee" would be read preferably as "For every manager m there is some employee e such that m manages e." Similarly, the preferred interpretation of "Some employee is managed by every manager" gives "some" wider scope than "every." Another heuristic, suggested by Hintikka [42] and used by Hendrix [40], associates with each determiner not only a corresponding quantifier, but also a "strength." Interpretations in which stronger

quantifiers outscope weaker ones are preferred. Thus "each" is stronger than "all," "any" and "some," so that in "Some manager manages each employee" there is a different manager for each employee, while in "Some manager manages every employee," either interpretation is possible, since "some" and "every" have similar strengths.

Presuppositions also affect scope. For example, in "What is the salary of all employees?" the determiner "all" probably should be given wider scope than "the," simply because it is unlikely that all employees would be receiving the same salary; the latter interpretation would violate the presupposition that the question has an answer. Although some computational work on presupposition has been done [101], [53], it does not deal with scoping.

Woods [110] proposed a compositional method for semantic interpretation in which phrases are assigned interpretations consisting of two constituents: a quantifier and a matrix proposition. The composition rules for a constituent combine the interpretations of the subconstituents by combining the matrix elements, nesting the quantifiers among themselves, or wrapping them around the matrices. This framework has been the basis for most scoping schemes since then. It has also been arrived at independently by theoretical linguists [21]. Woods's rules in LUNAR produce only one scoping, which is obtained by pushing quantifiers up the parse tree past their weaker counterparts until they reach a "hard" boundary, such as the top of a relative clause or a conjunction. Arbitration between quantifiers of similar strength is done on the basis of the left-to-right heuristic. A similar strategy is used in CHAT-80. TEAM applies a generate-and-test algorithm, in which all scopings that are not disallowed by syntactic constraints are produced; these are ranked by a set of heuristics. This framework allows better use of the quantifier strength heuristics.

In practice, the treatment of quantifier scoping in semantic-grammar systems is very limited. They could use LUNAR-style rules, but tend not to. Lacking an intermediate representation, they have no way of applying more global scoping strategies.

5.3. Discourse-Level Interpretation

Users of an NLI are typically interested in getting information from a database to use in solving some problem. It is rare that a single piece of information is all that is required; even when such is the case, the user may not be able to request it in a single query. Although no NLI contains a sophisticated or general model of the query dialogue, most incorporate some capabilities for handling a limited range of these discourse-related expressions. Special attention has been paid to some kinds of referring expressions (pronouns) as well as to certain constrained uses of elliptical phrases. In this section, we describe the basic techniques used in NLIs and provide a brief overview of

the techniques currently being investigated by researchers concerned with more general applications of NLP.

5.3.1. The Interpretation of Referring Expressions

Two kinds of referring expressions are prevalent in database queries, pronouns (especially, "it" and "they," but also "he" and "she") and definite descriptions ("the shoe department," "the U.S. peak"). To handle such expressions in a comprehensive manner requires a general model of the discourse context that takes into account the structure of the overall discourse and the purposes behind it [33, 64]; in addition, the model must take into account the features of the immediate discourse context of neighboring utterances [91, 32] as well as the structure and interpretation of an individual utterance [100, 37]. Each of these aspects of discourse context constitutes an active area of investigation in NLP.

The techniques used in NLIs are aimed not at providing a general solution, but at covering the most common uses of pronouns in database querying. Typically, the interpretation of pronouns is based on a "history list" which contains a record of the most recent preceding queries (i.e., some given number of these). The list distinguishes those expressions in each query that either introduce something new into the discourse or refer to something already introduced (these usually correspond to noun phrases), along with their interpretations and positions in the parse. When a pronoun is encountered, a search is made through the list (starting with the most recent entries) to find an expression or interpretation (depending on the type of system) that matches the pronoun (the same number and gender) and is compatible with the interpretation of the query.

For example, following the query "What is the division of the highest paid secretary?" the history list would include both "division of the highest paid secretary" and "highest paid secretary" (perhaps along with other information about each phrase). In interpreting the subsequent query "How many employees does *it* have?", the pronoun "it" is taken to refer to the same thing as "the division of the highest paid secretary" because divisions have employees and secretaries normally do not.

In semantic-grammar systems there are usually special rules that explicitly mention pronouns. For example, the following pair of rules might be used to provide an interpretation of the query "What is its revenue?" following the query, "What department has the smallest number of employees?"

```
<SENTENCE> → what is <DEPT-POSSESSIVE> <ATTRIBUTE>
<DEPT-POSSESSIVE> → its
```

When a pronoun is encountered in a particular construction, one of these rules is matched. This triggers a search through the history list for an expression that matches a particular category; the category searched for depends on the matched rule.

LUNAR also allows for references to objects dependent on other quantified objects, as in "What is the silicon content of each volcanic sample? What is *its* magnesium content?" The most general treatment of pronouns in IRL systems takes into account the syntactic structure of preceding queries to give a preference ordering on candidates and omit certain of these on the basis of syntactic constraints [45]. Various aspects of the pronoun resolution problem have been treated more generally in NLP research; Hirst [43] provides a good overview.

Because an adequate treatment of definite descriptions requires a model of discourse context, NLIs typically ignore the referring properties of such descriptions and take their interpretation to be all objects matching the description. In essence, these systems assume either a particular context in which there is only one object that matches a certain description or they assume that all items fitting that description are equally relevant. They ignore the difference between definitely and indefinitely determined noun phrases (e.g., "the G.M. employees" and "G.M. employees" are treated identically). Although this may be fine for an isolated query, it can lead to incorrect responses in context. For example, in isolation the query, "Who manages the G.M. employees?" might be a request for a list of the managers of all G.M. employees; on the other hand, in a context in which the user has just asked for the names of all employees earning more than $30,000, it may be a request solely for the managers of those G.M. employees earning more than $30,000.

5.3.2. Ellipsis

The term **ellipsis** refers to the omission of certain elements from what would ordinarily constitute the full syntactically correct form of a phrase. The interpretation of an elliptical phrase depends on recovering the missing information from the context in which the phrase is used. The treatment of ellipsis in NLIs has been restricted to the use of elliptical queries like those given in the beginning of this paper.

Two different approaches to ellipsis have been taken. One is to encode elliptical phrases directly in the grammar; the other is to modify the parser. The second approach not only allows broader coverage, but also is more easily adaptable to new domains and databases.

The encoding of elliptical fragments directly in the grammar has been done both for IRL systems [97] and for semantic-grammar systems [11]. In each case special grammar rules provide for incomplete phrases to be used in certain circumstances. For example, a syntactic grammar might include a rule like

$$S \rightarrow NP$$

to allow a single noun phrase to be used in place of a complete sentence. Likewise a semantic grammar might include a rule such as

<query> → <division>

Such rules would cover a sequence like

Who are the secretaries in the sales department?
The research department?

The interpretation rules or processes attached to these fragment rules construct an interpretation of the fragment and then search through the history of previous interactions (in some cases, only the preceding query is considered; this is often correct) to find an interpretation into which this piece can fit; the match is determined on the basis of a number of constraints, typically including lexical and encyclopedic ones.

A more general solution is provided by modifying the parser. This has been done for semantic-grammar NLIs that are based on a top-down parse using an ATN [38], but not for NLIs with more general grammars. The resulting parser remains efficient for the semantic grammars because of the additional semantic and pragmatic information encoded directly in them.

5.4. Semantic Coverage

One of the most important questions in NLIs is the relation between the expressivity of NL, IRL and QL. IRLs are less expressive than NLs, if only because their basic vocabularies (predicates and constants) are restricted to specific domains and tasks. They may, however, be more expressive than QLs in that they may admit logical concepts that are beyond the deductive abilities of the DBMS that interprets the QLs. The logical form of the TEAM system, for example, allows for modal operators (such as tense) and higher-order functions (such as maximum, count, and average) that lie beyond the deductive abilities of relational calculus, although their addition still leaves the QL decidable. This extra expressivity, often obtainable at little cost, makes it possible for parts of the NLI to be used eventually with software systems of greater deductive power.

There may be NL queries for which no corresponding QL representations exist. However, we claim that for any query that can be put to a DBMS in QL, there should be a corresponding query in NL that the NLI can translate into QL to generate the same answer. We call this the **accessibility requirement.** It is the analogue in NLIs of Turing equivalence between a high-order programming language and the language into which it is compiled.

In the remainder of this section we show that NLIs in general do not meet the accessibility requirement. In the following section, we illustrate ways of regaining accessibility.

The translation from IRL to QL is usually done according to what we will call the

rewrite method: atomic elements of the IRL representation language are rewritten into possibly complex *expressions* of QL. Thus, for example, IRL atoms may be mapped into expressions in QL that contain references to various parts of the DB (files, fields, values, etc.) and operations upon them. In relational algebra, such operations would include union, projection, and join--often enhanced by the so-called aggregate functions, such as maximum, minimum, average, and count. In logic-based systems, the operators are those of first-order logic.

Any NL query representable in QL has an answer in the DB, as all relational-calculus queries are decidable. There are, however, NL (or IRL) queries to which there exist answers in the DB, but which have no corresponding QL queries, at least none constructible under the rewrite method assumption. For example, the Navy Blue File, for which the LADDER system was written, contained a SHIP file in which a Boolean field DOB (for doctor-on-board) recorded whether or not a ship carried any doctors. The database contained no other mention of doctors, or of persons being on board ships. Thus, the IRL concepts *doctor* and *on-board-of* cannot be expressed separately as relational-calculus expressions in this database. As a result, the query "Is there a doctor on board the Fox?" can be interpreted only if the phrase "a doctor on board" (or its IRL equivalent) can be rewritten directly into a reference to the database field DOB.[14]

Introducing special translations for fixed phrases does not solve the problem in general. For example, the query "Is there a doctor within 500 miles of the Fox?" can be answered from the information in the Blue File, but it can be interpreted only by introducing translations for *doctor* and *on-board-of* separately.

The problem is not that the information is lacking in the database; that would explain why the query "How many doctors are on the Fox?" could not be answered. Neither is it only that the database does not represent certain objects, properties, and relationships directly (e.g., the Blue File does not explicitly represent doctors, or indicate who is on what vessel), and that it is not possible, by means of relational algebra, to construct from the existing relations one that does represent these explicitly (doctors, for example). The problem is inherent in the assumption of the rewrite method that atoms of the IRL map to expressions in the QL; hence, this method does not provide a way to take expressions in IRL to atoms in QL. The deductive method described in the following section is one solution.

---

[14] A similar problem arises in a database in which every person is related directly to his or her grandfather, e.g., in the single relation GRANDFATHER(YOUNG, OLD). The query "Who is the father of the father of John?" has an answer in the DB, but "father" is not expressible as a function of GRANDFATHER.

6. Future directions

Thus far, we have focused our attention on natural-language interfaces to DBMS. More broadly, in the context of natural-language processing, it is important to consider what issues need to be addressed to provide capabilities for users to communicate in natural language with a wider range of software. Two major obstacles stand in the way.

- Providing general procedures for bridging the gap between the concepts that can be expressed in natural language and the underlying software systems.

- Providing general mechanisms to allow the user and the computer system to cooperate in solving the user's problem by engaging in a dialogue.

One strategy for overcoming the first obstacle is suggested by a solution to the problem inherent in the use of the rewrite method--i.e., that certain queries that can be made in QL cannot be asked in NL. Instead of placing the semantic burden on the QL, as most existing systems do, this strategy places it on the IRL.

The ability to sustain interaction requires a different perspective as to the function of the interface. It must be considered not merely as a translator of sentences of one language into those of another, but rather as a recognizer of the user's intentions and as a collaborator in bringing about their satisfaction.

6.1. Putting Query Languages in their Place

A solution to the doctor-on-board problem is readily available if two conditions are met: (1) first-order logic (FOL) is taken as as the IRL, and (2) all the information in the database is encoded in IRL. The second condition can be relaxed as we will do shortly. Under these assumptions, it is now possible to define the relations encoded in the DB directly in terms of the domain concepts in IRL, rather than vice versa. If the contents of the DB are now converted into ground literals in IRL, the answer retrieval process can be implemented as deduction in IRL. In the ship DB, this means including an axiom that defines the DOB field from the DB in IRL:

$$\texttt{DOB(x)} \Leftrightarrow \exists \texttt{d ship(x)} \wedge \texttt{doctor(d)} \wedge \texttt{on-board(d,x)}$$

where *ship*, *doctor*, and *on-board* are predicates of IRL. The query "Is there a doctor on board the Fox?" would be represented in IRL by

$$\exists \texttt{d} \; \exists \texttt{x ship(x)} \wedge \texttt{doctor(d)} \wedge \texttt{on-board(d,x)} \wedge \texttt{x = Fox}$$

which is true if DOB(Fox) is true. Similarly, "Is there a doctor within 500 miles of the Fox?" would be represented in IRL by

$$\exists \texttt{d, dloc, sloc, s, dist doctor(d)} \wedge \texttt{location(d,dloc)} \wedge$$
$$\texttt{location(s,sloc)} \wedge \texttt{s = Fox} \wedge$$

distance(dloc, sloc, dist) ∧ dist < 500 miles.

Obtaining the correct answer now depends on having axioms such as

on-board(d,x) ∧ location(d, dloc) ∧ location(s, sloc) ⇒dloc =

We will call this second view of the language-to-DB correspondence the **deductive method.**

Now, in a sense, the deductive method is an unacceptable solution to the answer retrieval problem, because it does not use the DBMS as an inference engine--all deduction is done directly in IRL. Konolige [57] presents a better solution in which a QL query is actually constructed, but deduction rather than rewriting is used. The language in which deduction is performed contains IRL, but it also includes as *terms* the syntactic constructs of QL. Axioms are provided that express the relationships between the relations of IRL and the terms of QL.

Konolige's solution suggests a picture of the relation between an NLI and its underlying software that is rather different from the one suggested by analogy to programming-language compilers. The NLI must be able to draw inferences on its own, independently of whatever "black boxes" it may be connected to. Some of these boxes may themselves be specialized inference machines (DBMSs are clear examples of this), but their operation and semantics must be subordinate to those of NL.

6.2. Participating in a Dialogue

Although superficially it may appear that users of NLIs are merely asking questions, at a deeper level they are almost always engaged in a problem solving activity that requires them to obtain information from the DB. The view that interactive sessions with NLIs are instances of cooperative problem-solving behavior offers a more useful perspective not only on interaction with a database in particular, but on human-machine interaction in general. From this perspective, a user is seen as interacting with a system to effect a certain change in the world. The user might intend to accomplish this directly by getting the system to *do* something, or indirectly by getting the system to *communicate* some fact. Utterances are actions that change the world and provide information about the mental state of the utterer--most notably, about certain of his or her beliefs and intentions [4, 88].

When language use is examined from this perspective, discourses (i.e., extended sequences of utterances), not individual utterances are the natural unit of analysis; what the user intends to do and not what he has said is ultimately what matters. This point of view may make a difference even for some simple database query applications (the need to take this view can be inferred somewhat from the range of constructions that most NLIs attempt to handle and that go beyond simple questions), but it is vitally important from the standpoint of providing NL interaction with a broader range of

software systems (e.g., decision support systems). This point is nicely illustrated by the following short dialogue segment:

(1) U: I need to know which divisions earned less than $500,000 in 1985.

(2) S: The automobile division

(3) U: Consider its performance over the last five years.
(4) Can you show me a histogram by month?

Although Utterance (1) is superficially a statement about U's mental state, it is intended as a request for some information. If it were merely a report on U's mental state, a response acknowledging that (e.g., "OK. I understand.") would then suffice but such a reply is clearly unreasonable. Utterance (3) demonstrates that, even in a simple query-like context, the system's responses are an important part of the dialogue. The "its" is used to refer to the automobile division, a singular entity; Utterance (1) contains only a plural noun phrase and, if Utterance (2) were ignored, it would seem that there was no compatible prior phrase supplying a referent. Furthermore, the considering to be done depends on both Utterances (1) and (2). Utterance (3) is not about the domain of discourse, nor is it even a query, but rather about the discourse per se: it establishes a particular focus of attention for the discourse, namely, the performance of the automobile division over the last five years. Utterance (4) can be treated properly only by taking the context of the preceding utterances into account. What we have here is a request for a histogram of the monthly performance of the automobile division over the last five years. Finally, Utterance (4) is a request for a particular action to be taken; although ostensibly it asks for a "yes" or "no" response, neither of these would be adequate in and of itself; the "yes" requires that the system supply the histogram and the "no" obligates it to explain why it cannot do so.

Several areas of active research are concerned with devising methods for supporting NL communication on a broader basis. Some of this research is directly concerned with natural language; natural language provides both a set of particular problems to be addressed and a set of constraints on the theories being developed. Other research involves more general study of theories and models of purposeful action, but is nonetheless very relevant to work in NL. Activities in the following areas are of particular interest:

1. The connection between language and action: recognizing what a user intends (to do or have done) from what he says, as well as generating utterances that satisfy various intentions [18, 1, 17, 64, 3].

2. The connection between the intentions of individual utterances and the overall purpose of a discourse [47, 33].

3. Interactions among beliefs, desires, intentions, actions and plans [73, 71, 9, 58, 27].

These issues are of interest to a broad range of intellectual communities: theoretical computer science (because of their relevance to distributed computing systems), artificial intelligence (with its long-standing interest in machine reasoning and planning), the philosophy of mind (especially practical reasoning), and the philosophy of language (in which speech acts and reference are of central concerns). There continues to be much more to language understanding than language.

Acknowledgments

This paper appears in *Annual Review of Computer Science, Volume 1*, J. F. Traub (ed.), Palo Alto, CA: Annual Reviews Inc. and is reproduced here with the permission of the publisher. Its preparation was supported by the Defense Advanced Research Projects Agency under Contract N00039-84-K-0078 with the Naval Electronic Systems Command. We thank Martha Pollack and Jane Robinson for comments on earlier drafts.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States government.

References

1. Allen, J., Perrault, C.R. "Analyzing Intention in Utterances". *Art. Int. 15* (1980), 143-78.

2. Appelt D.E. TELEGRAM: a grammar formalism for language planning. Proc. of Int. Joint Conf. on Art. Int., 8th, IJCAI, Karlsruhe, 1983, pp. 595-599.

3. Appelt, D. "Planning English Referring Expressions". *Art. Int. 26*, 1 (April 1985), 1-33.

4. Austin, J.L.. *How to do things with words.* Oxford University Press, London, 1962.

5. Bates, M, Bobrow, R.J. A transportable natural language interface. Proc. of 6th Ann. Int'l SIGIR Conf. on Research and Development in Information Retrieval, ACM, 1983.

6. Bobrow, R.J., Webber, B.L. Knowledge Representation for Syntactic/Semantic Processing. Proc. of Ann. Nat. Conf. on Art. Int., 1st, Am. Ass. for Art. Int., 1980, pp. 316-23.

7. Bobrow, D. and the PARC understander group. "GUS-1, a Frame driven Dialog System". *Art. Int. 8*, 2 (April 1977), 155-173.

8. Brachman, R.J., Bobrow, R.J., Cohen, P.R., Klovstad, J.W., Webber, B.L., Woods, W.A. Research in Natural Language Understanding - Ann. Report: 1 Sept 78 - 31 Aug 79. Tech.Rep. 4274, Bolt Beranek and Newman Inc., Cambridge, MA, 1979.

9. Bratman, M. "Two faces of intention". *Phil. Rev. 93*, 3 (1984), 375-405.

10. Bruce, B.C. "Case systems for natural language". *Art. Int. 6*, 4 (1975), 327-60.

11. Burton, R.R. & Brown, J.S. Toward a Natural-language Capability for Computer-Aided Instruction. In *Procedures for Instructional Systems Development*, H. O'Neil, Ed., Academic Press, New York, 1979, pp. 273-313.

12. Burton, R.R., Woods, W.A. A Compiling System for Augmented Transition Networks. COLING 76, Int'l Conf. on Comp. Ling., 6th, Ottawa, 1976.

13. Charniak, E. Jack and Jane in Search of a Theory of Knowledge. Proc. Int'l Joint Conf. on Art. Int., 3rd, IJCAI, Stanford CA, 1973, pp. 337-43.

14. Chomsky, N.. *Aspects of the Theory of Syntax.* Massachusetts Institute of Technology Press, Cambridge, Massachusetts, 1965.

15. Codd, E. F. "A relational model for large shared data banks". *CACM 13*, 6 (1970), 377-87.

16. Codd, E. F. Relational completeness of data base sublanguages. In R. Rustin, Ed., *Data Base Systems*, Prentice-Hall, 1972, pp. 65-98.

17. Cohen, P.R., Levesque H.J. Speech acts and rationality. Proc. of Ann. Meet., 23rd, ACL, Chicago, 1985, pp. 49-60.

18. Cohen, P.R., Perrault, C.R. "Elements of a plan-based theory of speech acts". *Cognitive Science 3* (1979), 177-212.

19. Colmerauer, A. Metamorphosis grammars. In *Natural Language Communication with Computers*, L. Bolc, Ed., Springer-Verlag, 1978.

20. Colmerauer, A. "Un sous-ensemble intéressant du Francais". *RAIRO 13*, 4 (1979), 309-36.

21. Cooper, R. Variable binding and relative clauses. In *Formal Semantics and Pragmatics for Natural Language*, Guenthner F. and Schmidt S.J., Eds., Reidel, Dordrecht, 1979.

22. Culy, C.D. "The complexity of the vocabulary of Banbara ". *Ling. and Phil. 8* (1985), 345-51.

23. Dahl, V. "Translating Spanish into logic through logic". *AJCL 7*, 3 (1981), 149-64.

24. Damerau, F.J. "Operating statistics for the Transformational Question Answering System ". *AJCL 7*, 1 (1981), 30-42.

25. Davidson J., Kaplan, S.J. "Natural language access to databases: interpreting update requests". *AJCL 9*, 2 (1983), 57-68.

26. Earley, J. "An efficient context-free parsing algorithm". *CACM 13*, 2 (1970), 94-102.

27. Fagin, R., Halpern, J.Y. Belief, awareness, and limited reasoning. Proc. of Int. Joint Conf. on Art. Int., 9th, IJCAI, Los Angeles, CA, 1985, pp. 480-90.

28. Fillmore, C.J. *Syntax and Semantics*. Volume 8: The case for case reopened. In *Grammatical Relations*, Cole, P., Sadock, J.M., Eds., Academic Press, New York, 1977, pp. 59-81.

29. Finin. Constraining the interpretation of nominal compounds in a limited context. In *Analyzing Language in Restricted Domains*, Erlbaum, Hillsdale, NJ, 1985.

30. Gazdar, G., Klein, E., Pullum, G.K., and Sag, I.. *Generalized Phrase Structure Grammar*. Blackwell, 1985.

31. Ginsparg, J. A robust portable natural language data base interfce. Proceedings of Conference on Applied Natural Language Processing, ACL, 1983, pp. 25-30.

32. Grosz, B.J., Joshi, A.K., Weinstein, S. Providing a Unified Account of Definite Noun Phrases in Discourse. Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, June, 1983.

33. Grosz, B.J. and Sidner, C.L. The Structures of Discourse Structure. In *Discourse Structure*, L. Polanyi, Ed., Ablex Publishers, Norwoods, NJ, 1986.

34. Grosz, B.J., Appelt, D.E. Martin, P. and Pereira, F. "TEAM: An experiment in the design of transportable natural-language interfaces". *to appear in Art. Int.* (1986).

35. Harris, L.R. ROBOT: a high performance natural language data base query system. Proc. of Int. Joint Conf. on Art. Int., 5th, IJCAI, Cambridge MA, 1977, pp. 903-4.

36. Heidorn, G.E. "Automatic programming through natural language dialogue: a survey". *IBM J. Res. & Dev. 20* (1976), 302-13.

37. Heim, I. *The semantics of definite and indefinite noun phrases.* Ph.D. Th., Uni. of Mass., 1982.

38. Hendrix, G. G. Human engineering for applied natural language processing. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, IJCAI, Cambridge, MA, 1977, pp. 183-191.

39. Hendrix, G., Sacerdoti, E., Sagalowicz, D. & Slocum, J. "Developing a Natural Language Interface to Complex Data". *ACM Trans. on Database Systems 3*, 2 (June 1978), 105-147.

40. Hendrix, G. Semantic Aspects of Translation. In *Understanding Spoken Language*, D. Walker, Ed., North-Holland, New York, 1978, ch. 7, pp. 193-226.

41. Herskovits, A.. *Space and the Prepositions of English.* Cambridge Univ. Press, London, 1986.

42. Hintikka, J.K.K. "Quantifiers vs. Quantification Theory". *Ling. Inq. 5* (1974), 153-177.

43. Hirst, G.. *Lecture Notes in Computer Science.* Volume 119: *Anaphora in Natural Language Understanding.* Springer-Verlag, New York, 1981.

44. Hirst, G. *Semantic interpretation against ambiguity.* Ph.D. Th., Brown Univ., 1983.

45. Hobbs, J. "Resolving Pronoun References". *Lingua 44* (1978), 311-338.

46. Hobbs J.R. Selective inferencing. Proceedings of 3rd Conference, CSCSI, Victoria, B.C., 1980, pp. 101-122.

47. Hobbs, J. and Evans, D. "Conversation as planned behavior". *Cognitive Science 4*, 4 (1980), 349-377.

48. Hobbs, J.R. and Moore, R.C.. *Formal Theories of the Commonsense World.* Ablex, Norwood, NJ, 1985.

49. Hopcroft, J.E. and Ullman, J.. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley, Reading MA., 1979.

50. Isabelle, P. Another look ar nominal compounds. Proceedings of COLING-84, ACL, 1984, pp. 509-516.

51. Johnson, T.. *Natural language computing: the commercial applications.* Ovum, London, 1985.

52. Kaplan R. and Bresnan, J. Lexical-functional grammar: a formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*, J. Bresnan, Ed., MIT Press, Cambridge MA, 1982, pp. 173-281.

53. Kaplan, S.J. "Cooperative responses from a portable natural language query system". *Art. Int. 19*, 2 (1982), 165-88.

54. Kay, M. Functional Grammar. Proceedings, Berkeley Linguistic Society, Vol. 5, Berkeley, CA, 1979.

55. Kay, M. Algorithm schemata and data structures in syntactic processing. Nobel Symposium on Text Processing, Nobel Institute, Gothenburg, 1980.

56. Kay, M. Parsing in Functional Unification Grammar. In *Natural Language Parsing*, Dowty, D.R., L. Karttunen and A. Zwicky, Ed., Cambridge University Press, Cambridge, England, 1985, pp. 251-278.

57. Konolige, K. The database as model: a metatheoretic approach. SRI International, 1981.

58. Konolige, K. *A Deduction Model of Belief and its Logics.* Ph.D. Th., Stanford University, 1984.

59. Koskenniemi, K. *Two-level Model for Morphological Analysis.* Ph.D. Th., Univ. of Helsinki, 1983.

60. Kuno, S., Oettinger, A. Multiple Path Syntactic Analyzer. Information Processing 62, 1962, pp. 306-312.

61. Landsbergen, S.P.J. Syntax and formal semantics of English in PHLIQA1. Proc. of Int'l. Conf. on Comp. Ling., 6th, COLING, Ottawa, 1976.

62. Lehnert, W. G., Shwartz, S. P. EXPLORER: a natural language processing system for oil exploration. Proc. of Conf. on Applied Nat. Lang. Processing, ACL, 1983.

63. Lesser, V.R., Fennell, R.D., Erman, L.D., Reddy, D.R. "Organization of the Hearsay II Speech Understanding System". *IEEE Trans. Acoustics, Speech, and Signal Processing 23"*, 1 (1975), 11-24.

64. Litman, D.J. *Plan Recognition and Discourse Analysis: an Integrated Approach for Understanding Dialogues.* Ph.D. Th., Univ. of Rochester, 1985.

65. Marcus, M.P.. *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, MA, 1980.

66. Marcus, M. P., Hindle, D., Fleck, M. M. D-Theory: Talking about Talking about Trees. Proc. of Ann. Meet., 23rd, ACL, Cambridge, MA, June, 1983, pp. 129-36.

67. McCord, M.C. Modular Logic Grammars. Proc. of Ann. Meet., 23rd, ACL, Chicago, 1985, pp. 104-17.

68. McDonald, D. "Description Directed Control". *Computers & Mathematics 9*, 1 (1983), 111-130.

69. Montague, R. The proper treatment of quantification in ordinary English. In *Approaches to Natural Language: Proc. of the 1970 Stanford Workshop on Grammar and Semantics*, Hintikka, J.K.K., Moravcsik, J., and Suppes, P., Eds., Reidel, Dordrecht, 1973, pp. 221-242.

70. Moore, R. Problems in Logical Form. Proc. of Ann. Meeting, 19th, ACL, Stanford CA, 1981, pp. 117-124.

71. Moore, R.C. A formal theory of knowledge and action. In *Formal Theories othe Commonsense Word*, Hobbs, J.R., Moore, R.C., Eds., Ablex, 1985, pp. 319-358.

72. Moore, R.C. Handling complex queries in a complex database. Artificial Intelligence Center, SRI International, Menlo Park, CA, 1979.

73. Nilsson, N.J.. *Principles of Artificial Intelligence.* Tioga Publishing Co., Palo Alto CA, 1980.

74. Palmer, M.S. Inference-driven semantic analysis. Proc. of Nat. Conf. on Art. Int., 4th, AAAI, Washington, 1983, pp. 310-313.

75. Paxton, W.H. A framework for speech understnding. In *Understanding Spoken Language*, D. Walker, Ed., Elsevier-North-Holland, 1978, ch. 2, pp. 17-120.

76. Pereira, F.C.N. "Extraposition grammars". *AJCL 7*, 4 (1981), 243-56.

77. Pereira, F.C.N. *Logic for Natural Language Analysis.* Ph.D. Th., Univ. of Edinburgh, 1983.

78. Pereira, F.C.N., Warren, D. "Definite Clause Grammars for Language Analysis". *Art. Int. 13* (1980), 231-78.

79. Pereira, F.C.N, Shieber, S.M. The semantics of grammar formalisms seen as computer languages. Proc. of Int'l Conf. on Comp. Ling., 10th, COLING, Staford CA, 1984.

80. Perrault, C.R. "On the mathematical properties of linguistic theories". *Comp. Ling. 10* (1984), 165-76.

81. Petrick, S.J. Transformational analysis. In *Natural Language Processing*, R. Rustin, Ed., Algorithmics Press, New York, 1973, pp. 27-41.

82. Robinson, J. "DIAGRAM: A Grammar for Dialogues". *CACM 25*, 1 (January 1982), 27-47.

83. Robinson, A. "Determining verb-phrase referents in dialog". *AJCL 7*, 1 (1982), 1-16.

84. Sager, N.. *Natural Language Information Processing* . Addison Wesley, New York, 1981.

85. Scha, R.J.H. Semantic types in PHLIQA1. Proc. of Int'l Conf. on Comp. Ling., 6th, COLING, Ottawa, 1976.

86. Schank, R.C.. *Conceptual Information Processing.* American Elsevier, New York, 1975.

87. Scott, D. Domains for denotational semantics. Proc. of ICALP-82, ICALP, Heidelberg, 1982.

88. Searle, J.R.. *Speech acts: An essay in the philosophy of language.* Cambridge University Press, Cambridge, 1969.

89. Shieber S. M. The design of a computer language for linguistic information. Proc. of Int'l Conf. on Comp. Ling., 10th, COLING, 1984, pp. 362-66.

90. Shieber, S.M. "Evidence against the context-freeness of natural language". *Ling. & Phil. 8* (1985), 333-43.

91. Sidner, C. Focusing in the Comprehension of Definite Anaphora. In , M. Brady, R. Berwick, Ed., MIT Press, Cambridge MA, 1983, pp. 267-330.

92. Slocum, J. A practical comparison of parsing strategies. Proc. of Ann. Meet., 19th, ACL, Stanford, CA, 1981, pp. 1-6.

93. Stonebraker, M., Wong, E., Kreps, P., Held, G. "The design and implementation of INGRES". *ACM Trans. on Database Systems 1*, 3 (1976), 189-222.

94. Tennant, H.R, Ross, K.M., Saenz, R.M., Thompson, C.W., Miller, J.R. Menu-based natural language understanding. Proc. Ann. Mtg., 21st, ACL, Cambridge, Mass., 1983, pp. 151-58.

95. Thompson, F. B., & Thompson, B. H. Practical natural language processing: The REL system prototype. In *Advances in Computers*, Rubinoff, M., & Yovits, M. C., Eds., Academic Press, New York, 1975, pp. 109-68.

96. Ullman, J. D.. *Principles of database systems.* Computer Science Press, Rockville, MD, 1982.

97. Walker, D.. *Understanding Spoken Language.* Elsevier North-Holland, New York, 1978.

98. Waltz, D.L. "An English language question answering system for a large relational database". *CACM 21*, 7 (July 1978), 526-39.

99. Warren, D.H.D, Pereira, F.C.N. "An efficient easily adaptable system for interpreting natural language queries ". *AJCL 8*, 3-4 (1982), 110-22.

100. Webber, B.L.. *A Computational Approach to Discourse Anaphora.* , New York, 1980.

101. Weischedel, R.M. *Syntax and Semantics.* Volume 11: A new semantic computation while parsing: presupposition and entailment. In *Presupposition*, C.K. Oh and D. A. Dinneen, Eds., Academic Press, New York, 1979, pp. 155-182.

102. Weischedel, R.M. and Sondheimer, N.K. "Meta-rules as a basis for processing ill-formed input". *AJCL 9*, 3-4 (1983), 161-77.

103. Wilks, Y. "An Intelligent Analyzer and Understander of English". *CACM 18*, 5 (May 1975), 264-274.

104. Winograd, T.. *Understanding Natural Language.* Academic Press, New York, 1972.

105. Winograd, T.. *Language as a cognitive process. Volume 1: Syntax.* Addison Wesley, Reading, Mass., 1983.

106. Wolf, J.J., Woods, W.A. The HWIM Speech Understanding System. In *Trends in Speech Recognition*, Lea, W.A., Ed., Prentice-Hall, Englewood Cliffs, 1980, pp. 1-24.

107. Woods, W.A. Semantics for a Question Answering System. Harvard University Computation Laboratory, 1967.

108. Woods, W.A. "Transition Network Grammars for Natural Language Analysis". *CACM 13*, 10 (Oct. 1970), 591-606.

109. Woods, W.A., Kaplan, R.M. and Nash-Webber, B.L. The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, MA, June, 1972.

110. Woods, W.A. Semantics and quantification in natural language question answering. In *Advances in Computers*, M. Yovits, Ed., Academic Press, 1978, pp. 1-87.

111. Woods, W.A. "Cascaded ATN Grammars". *Amer. J. Computational Linguistics 6*, 1 (1980), 1-15.

112. Younger, D.H. "Recognition and parsing of context-free languages in time n^3". *Inf. and Control 14* (1967), 189-208.

113. Zwicky, A.M., Friedman, J., Hall, B.C., Walker, D.E. The MITRE syntactic analysis procedure for transformational grammars. Proc. of Fall Joint Comp. Conf., AFIPS, 1965, pp. 317-326.