4

RADC-TR-90-7
In-House Report
January 1990

NATURAL LANGUAGE PROCESSING: A TUTORIAL (REVISED)

Sharon M. Walter

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RADC-TR-90-7 has been reviewed and is approved for publication.

APPROVED: [illegible]

SAMUEL A. DI NITTO, JR.
Chief, Command & Control Software Technology Division
Directorate of Command and Control

APPROVED: [illegible]

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command and Control

FOR THE COMMANDER:

[illegible]

IGOR G. PLONISCH
Directorate of Plans and Programs

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) RADC-TR-90-7	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Rome Air Development Center; 6b. OFFICE SYMBOL *(If applicable)* COES	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COES)
6c. ADDRESS *(City, State, and ZIP Code)* Griffiss AFB NY 13441-5700	7b. ADDRESS *(City, State, and ZIP Code)* Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center; 8b. OFFICE SYMBOL *(If applicable)* COES	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS *(City, State, and ZIP Code)* Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.
62702F	5581	27	30

11. TITLE *(Include Security Classification)*
NATURAL LANGUAGE PROCESSING: A TUTORIAL (REVISED)

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
In-House	FROM Jan 89 TO Jul 89	January 1990	102

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	05	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Natural Language Processing, Augmentent Transition Network, Parsing, ATN, Grammars, Artificial Intelligence, Conceptual Dependency Theory, Semantic Networks, Definite Clause Grammar

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This report is an introduction to Natural Language Processing (NLP) research, the subfield of Artificial Intelligence (AI) attempting to produce computer programs capable of communicating with humans in their own language (English, for instance). Definitions of AI terminology relative to NLP and an overview of Keyword, syntactic, and semantic parsing technologies are included. This report is an updated version of the RADC In-House TR-86-110, "Natural Language Processing: A Tutorial," by the same author.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL SHARON M. WALTER	22b. TELEPHONE *(Include Area Code)* (315) 330-3577; 22c. OFFICE SYMBOL RADC(COES)

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

TABLE OF CONTENTS

TABLE OF APPENDICES

I. Introduction

This report is a revised version of the RADC In-House Technical Report Number 86-110, "Natural Language Processing: A Tutorial" [Walter; 1986]. Changes from the original, in general, reflect advances made in the state-of-the-art in Natural Language Processing, particularly in language generation as well as in commercially-available interface systems.

The report is structured to serve as an entry level, practical guide to research in the field of Natural Language processing by computer. Its design and composition are meant to provide a good basis for further study and practice in the field.

Section II defines the varied goals of researchers involved in producing computer programs that `process' Natural Language and attempts to provide a feeling for the full challenge of the task by outlining the problems that require solution.

As a subfield of Artificial Intelligence (AI), describing Natural Language Processing often requires a subset of the terminology of AI. Section III of this report defines members of that subset in the context of the Natural Language processing task.

Various schemes for categorizing approaches to processing Natural Language input exist. The most referenced scheme, from Terry Winograd's influential book Understanding Natural Language [Winograd; 1972], partitions approaches into four groups based on their representation and use of knowledge: "special format" systems were the early, keyword and pattern matching systems; "text-based" systems had a prestored text and indexing schemes to retrieve pertinent data; "limited-logic" systems were those whose databases were stored in some formal notation and which retrieved information using simple logical inferences, and lastly; "general deductive" systems were those using a uniform notation for representing problems and knowledge. This report will divide Natural Language understanding technologies, basically, into `syntax-based' and alternative approaches. Section IV describes methods which are primarily `syntax-based'. Significant changes have been made to Section IV in order to make it easier to understand. Section V will describe research into alternative approaches, from simple pattern matching systems to more complex systems which attempt to work from, and develop, representations of meaning.

Section VI describes the evolution of, and the current state-of-the-art in, computer-generated Natural Language. A discussion of generation technology was absent from the original version of this report due to the early stage of research in that area. Natural

Language Generation (NLG) research has noticeably progressed and is receiving much more attention from AI researchers. Increased emphasis on NLG is largely a result of the need for explanation, to humans, of computer reasoning in Expert Systems as such systems progress toward field use.

Section VII, another section new to this report, is a discussion of commercially-available Natural Language software.

Appendices include a glossary of field-related terms. For each term, an index directs the reader to an explanation of the term in the report. The names of computer systems used in the report as examples of concepts are also listed in a glossary, each with page numbers of where the system is mentioned. None of the systems are described in detail.

Examples of concepts in the text are meant to demonstrate concept "basics": they may, or may not, be wholly descriptive of those concepts. Many examples have been taken directly (or with modifications for simplification) from previously published work. References are provided. Hopefully, the use and description of examples is faithful to the originating authors' perspective. The responsibility for incorrect or misleading interpretations of other published work rests solely with the author.

II. 'Processing' Natural Language

A. What Does It Mean to 'Process' Human Language?

In the present context, parsing a human utterance is transforming an input expression into a coded (computer language) representation of that expression for use in further processing. Most often, "parsing" implies creating a representation of the structure, or syntax, of an expression. Parsing can also refer to the creation of a representation of the 'meaning' of an expression. In either case, parsing entails restructuring input into a representation more suitable for a specific application. The opposing aspect of 'processing' language 'generates' natural language from code representing information that is to be relayed to a human. Processing natural language combines 'parsing' and 'generation' with additional manipulations to provide a computer with the facilities necessary to allow facile communication with humans--communication of the type that is typical of human-to-human communication. 'Additional manipulations' include a broad range of inference processes, that is, processing to determine the real, intended meaning of a human statement to a computer. Such processing would include attempting to interpret Speech Acts ([Searle, et al; 1980]), and incorporating knowledge about user Belief Models ([Bruce; 1983]) and the dynamics of human Discourse ([Grosz and Sidner; 1986]) into its interpretation. An excellent introduction to Speech Acts, Belief Models, and Discourse can be found in [Allen; 1987] but inclusion of discussion of them at any length here would extend the bounds of this report beyond its intended introductory/tutorial level.

B. Why Attempt Computer Processing of Natural Language?

Language Translation

The desire to automate language translation was the stimulus for research into computer understanding of human language in the early 1950's. Efforts were aimed at converting Russian into English in word-for-word language translations. An oft-repeated (although fictional) anecdote illustrates the inadequate results of word-for-word translation: The sentence, "The spirit is willing but the flesh is weak" when translated into Russian, and then back into English by a language translation program, became: "The vodka is strong but the steak is rotten." The point made is that vast amounts of knowledge are required for effective language translations.

The initial goal for Language Translation was "fully-automatic high-quality translation" (FAHQT). In 1966, a report of the results of a study by the Automatic Language Processing Advisory Committee (ALPAC) to the National Research Council harshly criticized Language Translation research that had taken place up to that point, and disparaged of there ever being any useful results. That report brought funding for machine translation projects in the United States to a virtual standstill. The content of that report can be found in [Bar-Hillel; 1960]. Much of the criticism was recanted in a later report ([Bar-Hillel; 1971]), but the damage had been done. To this day there is virtually no government funding for translation research in the US, although the Japanese and European governments are heavily subsidizing such work. More realistic expectations and the realization that Machine Translation that is less than "fully-automatic" can be very useful, have lead to the resurgence of interest in the latter countries. The revised expectations are appropriate in light of the fact that essentially all human translations are post-edited and revised.

Two separate areas within present-day automated language translation technology are Machine Translation (MT) and Machine-Aided Translation (MAT). An MT system is one that is solely responsible for the complete translation process. Preprocessing of source text or post-editing may be done, but during actual machine processing there is no human intervention. MAT systems divide into two groups: Human-Assisted Machine Translation (HAMT) and Machine-Assisted Human Translation (MAHT). HAMT systems are those which are responsible for producing the translation but may interact with a human monitor along the way. MAHT systems are those in which a human is on-line and responsible for the translation but may interact with the system, for example, to request dictionary assistance.

Examples of currently available language translation systems are included in Section VII.

[Slocum; 1985] provides an exceptional tutorial on language translation, describing terminology, research history, and systems that are in use.

Natural Language Front-Ends to Databases and Expert Systems

Natural Language Processing systems cannot handle every possible phrasing of every possible statement, but they can be engineered to handle an amount sufficient to make themselves useful in many cases. NLP systems that ease communication to complex computer programs have been demonstrated. LUNAR [Woods; 1972], for example, answered questions about moon rock samples from the Apollo-11 mission using a large database provided by NASA. A more contemporary system is IRUS-86 [Weischedel, et al; 1989], which provides access to a large database of Navy information. Such interfaces usually work within a single, specific domain and are costly to develop. The goal then, is to create interfaces that can be customized to be useful in various domains.

NLP technology has progressed to the point that some systems, designed to be tuned to user requirements, are commercially available. To date, most of those systems provide access to databases. Database query is constrained by the structure and contents of a database. The breadth of possible inputs is limited to simple requests to retrieve specific information from structured information bases.

Language processing becomes more difficult, however, when more fluent use of language, for a broader range of uses, is desired. For example, simple inferences that people make in everday language are not so simple for computers. Ask such a system if John Q. Public is a senior citizen. The system may have the explicit information that Mr. Public is 66 years old, but not be able to answer the query.

Some of the complexities of everyday human language are noted in Part D of this Section. Systems for custom-built Natural Language interfaces have become available in recent years (see Section VII).

Text Understanding

Text Understanding is the automatic processing of textual information. A Text Understanding system reads message narratives and produces a representation of the contents that can be used to produce summarizations for busy human operators, or, in other ways appropriately distributes the message contents. It differs from the task of NL front-ends to databases and Expert Systems because there is no human interacting with the system. Since there is no dialogue between the computer interface and the human user, there is no integrated means of correcting inputs to the system and clearing up misunderstandings.

Text (or Message) Understanding is most predominantly used in military domains. An example is the PROTEUS/PUNDIT system that extracts information from Navy CASEREPs (CASualty REPorts) and puts it into a database to be accessed by an Expert System ([Grishman and Hirschman; 1986]). CASREPs are a class of reports about shipboard equipment failures. They provide information about ship readiness and equipment performance.

Intelligent Computer-Aided Instruction (ICAI)

In contemporary technology, Intelligent Tutoring Systems (ITS) (or, Intelligent Computer-Aided Instruction (ICAI)) replaces the concept of Computer-Aided Instruction (CAI) as it was described in the original version of this paper. The difference, according to [Burns and Capps; 1988], is the demonstration by an ITS of the use of three types of knowledge. An ITS must have expert knowledge of the subject that is to be taught, must be able to deduce the students' knowledge of that subject, and must be able to implement strategies that will bring the students' level of knowledge up to its own expert level.

SOPHIE (SOPHisticated Instructional Environment) is a well-known, early example of a training system. SOPHIE provided instruction on electronics troubleshooting, presenting students with a malfunctioning piece of electronics equipment and a challenge to find the fault in the circuitry. The system randomly selected a fault, inserted it into a simulation of a circuit, and explained the control settings for the faulty circuit to the student. In dialogue with the student, SOPHIE answered questions, critiqued hypotheses, and made suggestions designed to develop the student's troubleshooting skills. Sample dialogue of an early version of SOPHIE interacting with a student is shown in Appendix B of this report.

One of the most well-known ITSs under development is PROUST. PROUST diagnosis nonsyntactic errors in students' Pascal programs. PROUST is an off-line tutor in that it does not interact, in dialogue, with a student. Complete student programs are submitted to PROUST, which provides a printed diagnosis.

An ITS called STEAMER provides a graphical simulation of a steam propulsion plant. Students are allowed to change various aspects of the state of the model in order to observe the effects of their changes.

The current state of ITS technology, the ITSs mentioned in this Section, and others, are described in [Polson and Richardson; 1988]. Because of the early state of development of Intelligent Tutoring Systems, there are no examples of marketed systems in Section VII of this report.

Cognitive Modeling

The goal of some researchers in attempting to develop computer programs that process Natural Language is to understand how humans produce and understand language. Their research attempts to 'model' the exact processes used by humans (cognitive modeling). Hypotheses are often tested using various experiments that compare specific aspects of human versus computer processing of language. Parsifal, described briefly in Section III, is an example of a computer program designed to model a specific aspect of human language understanding.

C. What Knowledge is Required for Processing Natural Language?

Largely, the amount and type of knowledge that must be incorporated into a program for language processing depends on the degree of understanding desired. Systems using simple pattern matching and word phrase substitution, as in those described by Section V.A. (Keyword/Simple Pattern Matching Systems), demonstrate no real language understanding but may be satisfactory in some instances. Most useful Natural Language processing will require a much greater degree of sentence analysis and understanding. Some will require understanding comparable to human understanding.

Comprehension of a sentence requires, as a minimum, an understanding of the syntax, semantics, and pragmatics of human language. The "syntax" of a language defines the 'structure' of objects in the language. It describes how basic units fit together to form structurally (syntactically) correct members of the language. In human language for example, if a sentence can be a sequence of words such that the first is from the syntactic class of adjectives, the second is an adjective, next a noun, a verb and, finally, an adverb, then the following is a syntactically correct sentence:

(1) Colorless green ideas sleep furiously. [Chomsky; 1965]

The Autonomy of Syntax Hypothesis proposes that syntax, as a distinct component of language, can be studied as a distinct entity, later defining its relationship to sentence semantics and pragmatics. Research in Computational Linguistics initially focused exclusively on the syntactic language component as it was presumed to be sufficient for the task of Machine Translation.

The "semantics" of a language associates a `meaning' to each of its constructs. The syntax of the following sentences is the same, but the semantics are obviously different:

(2) The artist painted using a new technique.
The artist painted using a new paintbrush.

"Pragmatics" is the study of language in context. It is the `intended meaning' of an utterance. Pragmatic knowledge can eliminate the "stonewalling behavior" demonstrated in the following exchange:

(3) Do you know Sarah's last name?
Yes.

Could you give it to me?
Yes.

Can you tell me Sarah's last name?
Yes.

Understanding idioms also requires the use of pragmatic knowledge. An idiom is a sequence of words with a special meaning when taken as a unit, apart from its literal meaning when taken word for word. To say that someone "kicked the bucket" has a literal meaning, but more often implies that a person has ceased to live.

In recent years there has been a surge in studies of language semantics and some, limited, study of pragmatics. The format of interaction among language components is somewhat controversial. In some systems, LUNAR for instance (described briefly in II.B.), a first pass through a sentence produced a representation of its syntactic structure and a second phase used the result to construct a semantic interpretation. Systems based on Roger Schank's Conceptual Dependency (CD) knowledge representation theory (See section V.C.2) build a semantic structure directly from the input string, using syntactic information when necessary. Terry Winograd's SHRDLU program [Winograd; 1972] attempted to deal with syntax and semantics in an integrated way. SHRDLU simulated a robot arm in a small domain containing blocks of different shapes and colors. In a dialogue with the system a person can give commands, state facts, and ask questions about the state of the block's world.

D. What Problems are Encountered When Attempting to Provide Natural Language Understanding Capabilities to Computers?

There are a number of aspects of human language that a system must handle in order to be capable of fluent communication. To take human imperfection into account, a system must understand misspelled and ungrammatical inputs. Additional features of natural language to consider are: ambiguity, anaphoric reference, and ellipsis.

A human utterance is ambiguous if it has two or more possible interpretations. There are a variety of reasons why a statement might be ambiguous. A word which belongs to more than one syntactic class may cause an utterance to be ambiguous. For instance, the word "duck", interpreted as a noun in "He saw her duck" (and "her" as an adjective) implies that an animal was seen. If "duck" is interpreted as a verb (and "her" as a noun), the sentence implies that an act of "ducking" was seen.

Some words have multiple senses, or meanings, which belong to the same syntactic class. The appropriate sense of the word must be determined by its context in the sentence or situation. A classic example is the use of the word "pen" in "The pen is in the box" and "The box is in the pen". There "pen" can refer to a writing instrument or a child's playpen, a noun in either case.

Sentences may be "structurally" ambiguous, as in:

(4) "Waiter, I would like spaghetti with meat sauce and wine."
"Waiter, I would like spaghetti with butter and garlic."
[Charniak; 1976]

In the first sentence, "spaghetti with meat sauce" and "wine" are understood as individual phrases. In the second, "spaghetti" and "butter and garlic" make up individual phrases. The structure of these sentences is made clear by their meaning. The structure of "the old men and women" is not so easily clarified. Does the phrase refer to old people of either gender, or does it refer to women of any age and men who are old? Conjunctive phrases, phrases joined by "and", and disjunctive phrases, those joined by "or", are a major cause of ambiguous syntactic structure. Structural ambiguity might also be caused by confusion over which sentence concept is being modified by a prepositional phrase. The classic example is: "I saw the man on the hill with the telescope." Who is on the hill, "the man" or the speaker? Who has the telescope, "the man", the speaker, or "the hill"?

Anaphoric reference is reference to something previously mentioned in the conversation without being specific in that reference. The following examples demonstrate the concept, and some of the distinguishable types, of anaphoric reference [Webber; 1979]:

(5) a. Pronomial Reference

As in: "Today I met a man with two heads. I found him very strange."

The pronoun "him" refers to the just-mentioned man with two heads.

b. Noun Phrase Anaphora

As in: "Today I met a man who owned two talented monkeys. The monkeys were discussing Proust."

"The monkeys" alludes to the two just-mentioned monkeys.

c. "One" Anaphora

As in: "Wendy got a blue crayon for her birthday and I got a purple one."

The "one" is a crayon.

d. Set Selection Anaphora

As in: "The vendor listed fifteen flavors of ice cream. Bob wanted the fifth."

(ie. "the fifth" flavor of ice cream)

Recent trends toward integrating modes of communication with computers (eg. typed input, graphic displays, touch screen input,...) have introduced a new, but similar problem of determining the object(s) of reference: deictic reference. Deixis involves reference to things that have not been explicitly referred to (by speech or typed input), but are present in some way in the nonlinguistic context of the conversation. For instance, a graphic display may be pointed to with a request to: "Identify this."

Ellipsis occurs in human conversation when people omit portions of sentences, assuming that the missing parts can be filled in by the listener (or reader). Consider this example:

(6) What kind of ice cream does Carol like?

$$\begin{tikzcd}
\text{Ted thinks} & {} \arrow[d] & \text{vanilla} & {} \arrow[d] & . \\
& \text{(that Carol likes)} & & \text{(ice cream)} &
\end{tikzcd}$$

E. What Representation Schemes are Used?

Many researchers attempt to form a computer language representation of input that stresses the syntax of the input. These representations can be displayed in graph forms similar to diagrams of sentences produced in grade school English class. Such a representation scheme is defined by a grammar.

A grammar of a language specifies the sequences of basic units that are allowed in the language. A grammar can be used to describe well-formed computer language constructs or well-formed human sentences. Grammars of languages, including computer languages, are described by a set of terminal and nonterminal symbols, and a set of rewrite (or production) rules. A distinguished nonterminal serves as the unique starting symbol for the grammar.

For the purpose of defining a grammar for English the non-terminal symbols are usually syntactic categories such as "Sentence", "Noun Phrase", etc. Nonterminals in this report will be abbreviated as S, NP, etc. The terminal symbols of natural languages are the words. Rewrite (or, production) rules specify the relationships among terminals and non-terminals. For example, S -> NP VP means that the symbol S can be rewritten, or replaced, by a NP followed by a VP. (Hence, a sentence is a noun phrase followed by a verb phrase.) Det -> "the", means the symbol Det can be rewritten by the terminal symbol "the".

To illustrate, here are the rules of a grammar for a very small subset of English:

(7a)	S -> NP VP	(7e)	Det -> the
(7b)	NP -> Det N	(7f)	N -> boy
(7c)	NP -> N	(7g)	N -> Mary
(7d)	VP -> V NP	(7h)	V -> kissed

The terminal symbols of the grammar are the words: "the", "boy", "Mary", and "kissed". The grammar indicates that these words belong, respectively, to the syntactic categories of determiner (Det), noun (N), noun, and verb (V).

Grammar rule (7a) defines a sentence (S) as a noun phrase (NP) followed by a verb phrase (VP). Rules (7b) and (7c) state that a noun phrase can be either a determiner followed by a noun,or simply a noun. Rule (7d) defines a verb phrase as a verb followed by a noun phrase.

Thus, the syntactic structure of the sentence, "The boy kissed Mary", may be represented in tree form as shown in Figure 1.

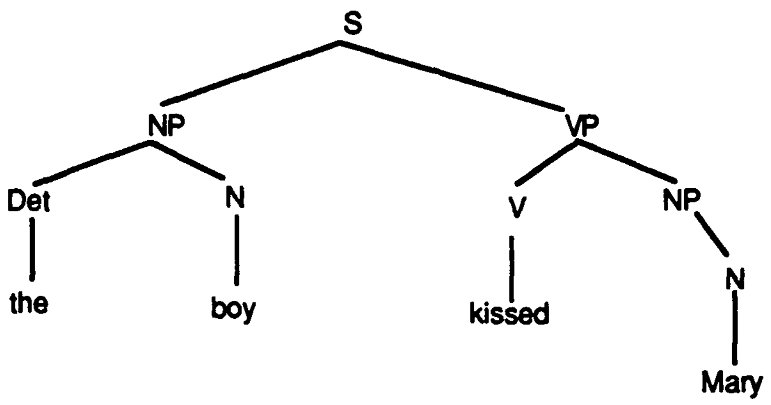


Figure 1
Syntactic Tree Structure of "The boy kissed Mary."

In reality, grammars designed to define human language contain some terminal nodes that are not words. These nodes are grammatical items (eg. "Past", "Possessive") that will later combine with other terminal nodes to determine the appropriate form of a word, for instance the past tense of a verb. Readers should be aware of this phenomena, but examples in this report for the sake of clarifying basic concepts will overlook this detail.

This report will categorize processing efforts which focus on the syntactic category of each word and the syntactic structure of each utterance as "syntax-based". Other researchers attempt coded representation of language using various other means and concentrating on other types of information (See Section V.).

III. Artificial Intelligence (AI) Terms Relative to Natural Language Processing

There are a number of terms, alluded to in the Introduction, having a general meaning in the broader context of Artificial Intelligence (AI) and more specific implications when applied to the field of Natural Language Processing. Those terms will be defined here.

The terms top-down and bottom-up processing are applicable to Natural Language Processing and are generally applied to syntax-based processing. Top-down or hypothesis-driven processing starts with the hypothesis that an input is a sentence and tries to prove it. Processing begins with the start symbol, S. A grammar rule to expand the S node is found and applied. At each succeeding stage, appropriate grammar rules are found and applied to the remaining non-terminal nodes until all non-terminal nodes have been expanded and the terminal nodes that remain are the words of the sentence being processed. The sample grammar of Section II will be used to clarify the notions of top-down and bottom-up processing.

Top-down processing begins with the start symbol S. Rule (7a) applies and produces this structure:

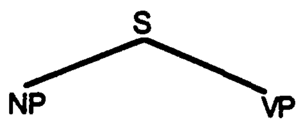


Figure 2

Rules (7b) or (7c) can expand NP. Rule (7d) can expand VP. In top-down processing, expanding the leftmost node first (in this case NP) and continuing to the right (left-to-right parsing) produces the same structure as right-to-left parsing. Thus at this point, the same structure results whether NP or VP is expanded first. Applying rules (7b) and (7d) will produce the structure shown in Figure 3.

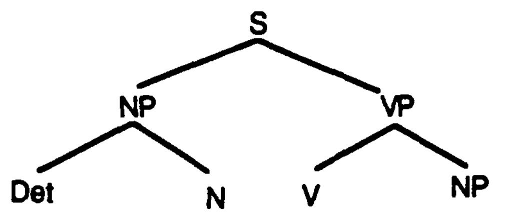


Figure 3

Note that a different structure, inappropriate for the current sentence being parsed, would be produced from expanding NP with rule (7c) rather than with rule (7b).

Applying rules (7e), (7f), (7h), and (7c), in any order, produces the structure in Figure 4. Finally, rule (7g) is applicable and produces the completed structure in Figure 1.

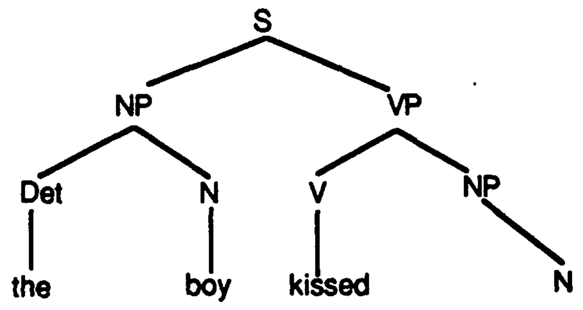


Figure 4

Bottom-up or data-driven processing begins with the input string. Using the grammar rules, each word is replaced by its syntactic category. When the right-hand side of a grammar rule is matched by an ordered string of these categories, the string is reduced by replacing it with the left-hand side of the rule. Categories are combined repeatedly until a single structure, with the sentence (S) category at the top, is reached. For our example sentence and grammar, the following structure is produced first (Figure 5):

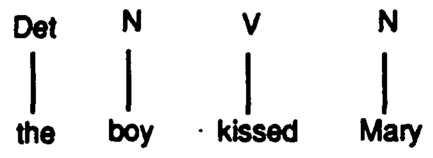


Figure 5

At this point, rule (7b) and rule (7c) can be applied to produce Figure 6:

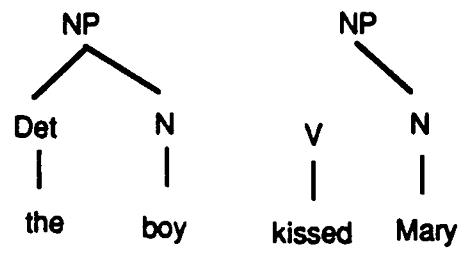


Figure 6

Rule (7d) combines V and the second NP into a VP (Figure 7).

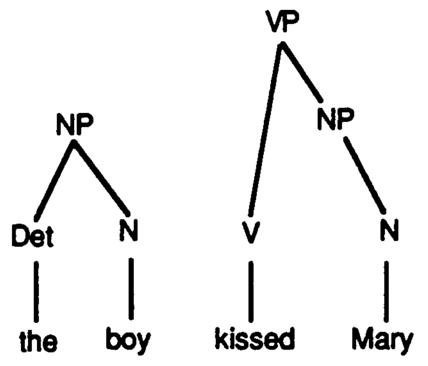


Figure 7

Finally, rule (7a) resolves the structure (Figure 1).

Even with this very simple grammar, there are occasions when two or more rules are applicable. If the applicable rules do not interfere with one another, a decision still must be made about the order of their application. In addition to left-to-right and right-to-left parsing, there is a third, less used tact for determining where processing will continue, called island-driving. Island-driving goes neither strictly from left to right, nor strictly from right to left. Rather, it uses specific heuristics, or rules-of-thumb, to direct which part of the input will be processed next and continues by processing neighboring chunks of increasing size. For example, in looking for a noun phrase a processor might look first for a noun, then look to the left of the noun for adjectives and determiners and look to the noun's right for modifying phrases.

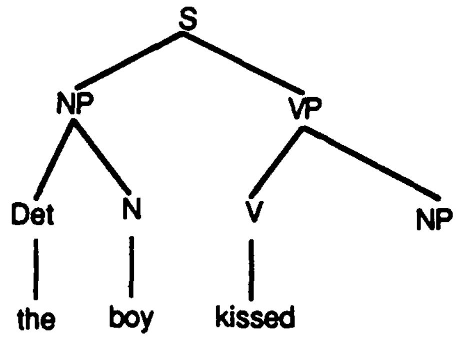


Figure 8

What do we do when two or more rules, applicable at some point in the parsing process, reference intersecting portions of the structure produced so far? Assume, for the sake of clarification, we are in the midst of a top-down parse of our sample sentence, "The boy kissed Mary", using the example grammar. Assume also that Figure 8 displays the current state of the parse. At this point, either rule (7b) or rule (7c) can be used to expand the NP. A deterministic parser would make a firm decision about which one of the applicable rules to use. After that rule is applied, the parser is committed to its decision, right or wrong, and will not get an opportunity to redo the parse in another way.

Mitchell Marcus contends that people parse language in a deterministic manner. Marcus designed a sentence parsing program, Parsifal [Marcus; 1980], to support this theory and measured the program's success at modeling human sentence understanding by comparing its competence with human competence in interpreting garden path sentences. Garden path sentences are sentences which tend to initially mislead readers, leading them "down the garden path", regarding their structure. A second, more conscious effort is required for understanding such sentences. The following are well-known examples of garden path sentences:

(8a) The horse raced past the barn fell.

(8b) The prime number few.[1]

[1][Milne; 1983] points out that Parsifal would understand (8b) without difficulty while human readers would usually garden path. Dr. Milne's parser, Robie, combines look-ahead with the use of non-syntactic information to resolve word sense ambiguities. Milne proposes that Robie more closely models human language processing, evidence by the fact that it garden paths on (8b).

Parsifal uses a technique called look-ahead to help decide on an appropriate parsing path. Parsifal holds off on making its decision and 'looks ahead' to get an idea about what is coming up before choosing its path.

A nondeterministic parser can deal with multiple paths using either depth-first or breadth-first processing. If a parser keeps track of all possible paths (or a useful subset of them) and continues building the structure on each of them in parallel, it is processing breadth-first (usually called parallel processing in the context of Natural Language processing). In depth-first processing one of the paths is followed. If, at some point, the choice is proven to have been wrong, the parser will back-up, or backtrack, to the state of the parse just previous to when the incorrect path was taken and try another path.

From the stage of the parse illustrated by Figure 8, if parallel processing is used, the following two structures would be built (Figures 9 and 10):

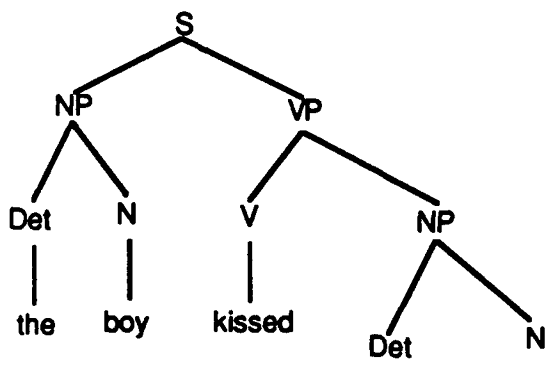


Figure 9

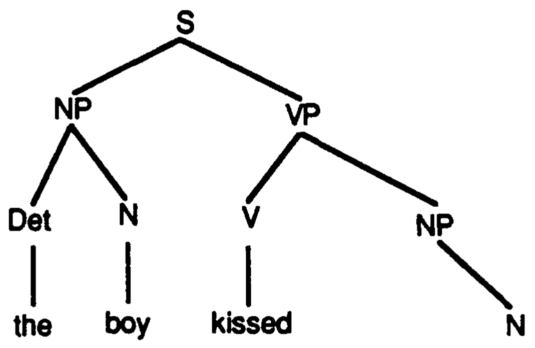


Figure 10

Figure 9 is constructed when rule (7b) is applied to the structure of Figure 8. Figure 10 follows from applying rule (7c) to the structure of Figure 8. As parallel processing continues, additional structure may be built onto each of these structures. In our particular example, (7e) is the only rule that will expand the symbol 'Det' in the structure of Figure 9, but it produces a structure with a terminal node that conflicts with the desired root nodes, the words of the sentence. Processing on the structure of Figure 9 halts, leaving Figure 10 as the only structure to build upon.

IV. Syntax-Based Processing

A. Linguistic Theory as a Basis for Processing Natural Language

A considerable percentage of Natural Language Processing research has developed from the scientific study of language (Linguistics). Some NLP systems are based directly on computer implementations of grammar theories. Thus, an overview of the basic ideas and history from Linguistic Theory is necessary background for an understanding of NLP technology as it is presented in this paper and for future study. Some of the theory and associated terminology of Linguistics is described here.

1. Linguistic Notation and Terminology

Specifying Grammars

In the notational system used in the following sections, lower-case letters a, b, c, d, and e will represent the basic units, or terminal symbols, of language. When describing a grammar for English, these terminals represent words. In the notation of Linguistic study, the letter T is used to represent the set of terminals of a language. So, for our notation, $T = \{a, b, c, d, e\}$.

Lower-case letters v and w will be used to denote strings of unspecified terminals.

Capital S will represent a unique "start" symbol. S and capitals A, B, C, D, and E will be variables, or nonterminal symbols. When describing a grammar for English, they represent syntactic categories such as Sentence, Noun Phrase, etc. In Linguistic notation, S is always used to represent the start state. The letter V is used to represent the set of variables for a language. So here, $V = \{S, A, B, C, D, E\}$.

Lower-case Greek letters α (alpha), β (beta), and χ (gamma) denote unspecified strings of variables and terminals.

As described in Section II.E., rewrite or production rules specify the relationships among terminals and variables. In Linguistic notation, P represents the set of production rules of a language.

To specify a grammar, one needs to identify its terminals, variables, unique start state, and production rules. So, to talk about a grammar, G, you must specify the members of the sets T, V, and P, and identify the start state. Such a grammar is denoted as $G = (V, T, P, S)$. All of the strings that can be generated using the grammar makes up its language, just as the grammar for English defines the "grammatically correct" English language. The language that is described by a grammar G (every statement that is grammatically correct according to the grammar) is denoted L(G).

"Normal" Forms for Grammars

There are algorithms for changing the production rules of a grammar into a standard or "normal" form. Normal forms for grammars allow them to be compared in order to test for equivalence, and the procedure for changing to the Normal form eliminates excesses in the grammar such as equivalent rules (or equivalent sets of rules) and rules that will never be used because their left-hand side will never appear in a string. Examples of Normal forms are the Chomsky Normal Form (CNF) and the Greibach Normal Form (GNF). An excellent reference for further information is [Hopcroft and Ullman; 1979].

2. Chomsky's Grammar Types

In the 1950's a linguist, Noam Chomsky, developed a classification system grouping grammars into classes according to the type of rules allowed in their definition. There are four types of grammars in this classification scheme, labeled Type 0, 1, 2, and 3. From Type 0 to Type 3, the rules of the grammars become more restrictive and the languages generated become simpler. Each grammar type, is a proper subset of the next lower-numbered type. Lower numbered grammars are considered "more powerful" because they allow a larger number of legal strings. Chomsky's classification is useful for describing grammars for computer or human languages.

Type 0 Grammars

A Type 0 grammar is one in which there are no restrictions on the form that the production rules can take. The left-hand side of each production rule describes a non-empty string of symbols. The right-hand side describes a string that may be used in place of any string that matches the rule's left-hand side. According to our notation, grammars of Type 0 have production rules of the form α -> β, where α and β are arbitrary strings of terminals and variables, with α a non-empty string. These grammars are also known as semi-Thue or unrestricted grammars.

Type 1 Grammars

A grammar is a Type 1 grammar if, for each production of the grammar the right-hand side consists of at least as many terminal and variable symbols as the left-hand side. Notationally, we say grammars of Type 1 have productions of the form α -> β such that α is non-empty and β is at least as long as α.

There is a Normal Form in which each production of a Type 1 grammar is of the form $\alpha A\chi$ -> $\alpha\beta\chi$, where β is a string of at least one symbol. This permits replacement of the variable A by a string of terminals and variables β only in the "context" α - χ. Type 1 grammars are often referred to as context-sensitive grammars.

An example of a context-sensitive grammar is the following (note that it is not in a Normal Form that demonstrates its context-sensitivity):

(9a) S -> aSBC
(9b) S -> aBC
(9c) CB -> BC
(9d) aB -> ab
(9e) bB -> bb
(9f) bC -> bc
(9g) cC -> cc

This grammar defines a language in which each string generated consists of a number of a's, followed by the same number of b's, followed by the same number of c's. The language generated cannot be defined by a grammar of a more restricted type (ie. Type 2 or Type 3).

The string `aaabbbccc' is a member of the language defined by this grammar and can be derived as shown below. The underlined symbols are those that are matched against the left-hand side of the next rule to be applied, and replaced by that rule's right-hand side.

	Rule Used	
Start Symbol:		$\underline{\mathrm{S}}$
	(9a)	$\mathrm{a\underline{S}BC}$
	(9a)	$\mathrm{aa\underline{S}BCBC}$
	(9b)	$\mathrm{aa\underline{aB}CBCBC}$
	(9d)	$\mathrm{aaab\underline{CB}CBC}$
	(9c)	$\mathrm{aaa\underline{bB}CCBC}$
	(9e)	$\mathrm{aaabbC\underline{CB}C}$
	(9c)	$\mathrm{aaabb\underline{CB}CC}$
	(9c)	$\mathrm{aaab\underline{bB}CCC}$
	(9e)	$\mathrm{aaabb\underline{bC}CC}$
	(9f)	$\mathrm{aaabbb\underline{cC}C}$
	(9g)	$\mathrm{aaabbbc\underline{cC}}$
	(9g)	$\mathrm{aaabbbccc}$

Type 1 grammars are said to have difficulty representing sentences that have similar meanings with similar structures:

(10) "John ate an apple."
"Did John eat an apple?"
"What did John eat?"

There are also cases where Type 1 grammars represent sentences with different meanings as having similar structures:

(11) "The picture was painted by a new technique."
"The picture was painted by a new artist."

Type 2 Grammars

Type 2 grammars are those in which each production rule has only a single variable on its left-hand side. In Type 2 grammars, production rules specify replacements for symbols without regard to what presently precedes or follows the symbol (i.e. the symbol's 'context'). These grammars, therefore, are context-free grammars.

The following is a Type 2 (context-free) grammar:

(12a)	S -> aB	(12e)	A -> bAA
(12b)	S -> bA	(12f)	B -> b
(12c)	A -> a	(12g)	B -> bS
(12d)	A -> aS	(12h)	B -> aBB

This grammar defines a language in which each string generated consists of an equal number of a's and b's. "bbabaa" is a string that is derivable from this grammar. Type 2 grammars are considered insufficient for describing natural, human language because they cannot produce strings of the form: $a_1\ a_2 \ldots a_n\ b_1\ b_2 \ldots b_n$, where each a_i corresponds in some way to b_i. Thus, Type 2 grammars cannot handle the use of the word 'respectively', as in:

(13) "Randy, Ken, and Charlotte are a doctor, a nurse, and a philosopher, respectively."

Type 3 Grammars

The production rules of Type 3 grammars are more restrictive than Type 0, 1, or 2 grammars. As with Type 2 grammars, the production rules for Type 3 grammars have only a single variable on its left-hand side. On the right-hand side, Type 3 rules consist of a single terminal symbol or a terminal symbol followed by a single variable. This most restrictive of the grammar types is called a regular grammar.

The simple example of a Type 3 grammar shown at the top of the next page generates strings of at least one a, followed by at least one b. The string "aaaabbb" is derivable from this grammar.

(14a) S -> aS

(14b) S -> aT

(14c) T -> b

(14d) T -> bT

The restrictions on Type 3 grammar rules prohibit production rules of the form B -> $\alpha B \beta$ where β is not necessarily empty. Such self-embedded constructs are presumed to be necessary in the context of Natural Language for dealing with center embedding.

In center embedding, a sentence is embedded *within* another sentence. Center embedding is the result of modification of a noun group that is not at the end of its encompassing sentence, by a clause with a sentence-like structure. A simple example is:

(15) "The rat that the cat chased ate the cheese."

The embedded sentence, "the cat chased the rat," is slightly changed in order to mark it as a noun phrase within the relative clause.

Often in normal language usage, the relative marker ("which", "who", "whom", or "that") marking the start of the relative clause is left out of the sentence, as in:

(16) "The rat the cat chased ate the cheese."

Center embedding can make sentences quite difficult to understand if the embedding is done more than once. However in principle they are grammatical. Embedding another sentence in the sentence shown in (16) is demonstrated by the sentence in (17), below.

(17) "The rat the cat the dog bit chased ate the cheese."

Another example of center embedding, from [Chomsky, 1965], is:

(18) "The man who the boy who the students recognized pointed out is a friend."

The individual sentences in this example of center embedding are: "The man is a friend." "The boy pointed out the man." "The students recognized the boy."

There is plenty of controversy about which class or classes of grammars are sufficient for describing natural language. The arguments, given above, against the applicability of Type 1, 2, and 3 grammars are quite well-known and widely held to be valid. Gerald Gazdar's arguments to the contrary can be found in any of the referenced materials by Gerald Gazdar (see reference list at the end of this report) and [Sampson; 1983]. (Also, see [Pullum; 1984].)

Type 1 and Type 2 grammar rules break sentences into a hierarchy of smaller and smaller constituents. They can be used to display the structure of a sentence as a tree, with each `node' expanding into one or more branches (never drawing nodes together into a smaller number of symbols). Each node of the tree and the structure below it represent a syntactically significant part of the sentence called a phrase marker, or p-marker. For example, the syntactic structure of "The cat likes the mouse" may be displayed by:

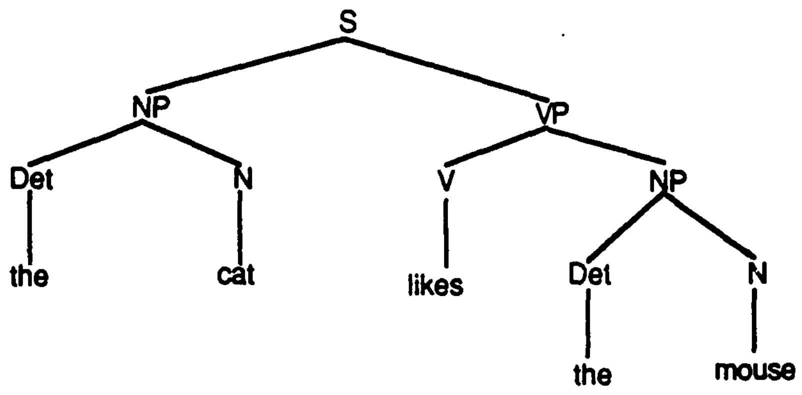


Figure 11
Syntactic Structure of "The cat likes the mouse."

Here, phrase markers predict that "likes the" is not syntactically significant, but "the cat" and "likes the mouse" are.

Because they display the "phrase structure" of sentences, Type 1 and Type 2 grammars are called phrase structure grammars. Phrase structure grammars are also referred to as immediate constituent grammars, indicating that each node is an "immediate constituent" or "child" of the node above it in the tree.

3. Computational Complexity Theory

A parsing algorithm is a set of rules that specifies which grammar rule, from among those that are applicable, should be applied next. Parsing algorithms are often compared to determine which is best, based on the amount of time and space required to parse inputs. Time is the most commonly measured resource. Measurement of the time required for parsing is a very rough estimate, measured as the maximum number of production rules needed to parse any input string of length n, where n is a variable. Measuring the time efficiency of an algorithm allows it to be put into a very broad class, a "complexity class", of algorithms with similar parsing efficiency. As an example, a parsing algorithm may have a complexity of n^2 (often presented notationally as $O(n^2)$ and read, "of the order n-squared"). That is to say, there is some number C such that Cn^2 is an upper bound on the number of steps required to parse any string of length n.

A number of parsing algorithms for context-free grammars are described in literature discussing computational complexity. Two of the most well-known are the Cocke-Younger-Kasami (CYK) algorithm and the Earley algorithm. The CYK algorithm is $O(n^3)$. The Earley algorithm, considered the most practical algorithm, is $O(n^3)$ in general, but only $O(n^2)$ on any *unambiguous* context-free grammar.

Recommended references for further information on Computational Complexity Theory and parsing algorithms are [Barton, et al; 1987], [Earley; 1970], [Hopcroft and Ullman; 1979], and [Sippu and Soisalon-Soininen; 1988].

4. Transformational Grammar

Transformational Grammar Theory was introduced by Noam Chomsky in Syntactic Structures (1957) and revised in Aspects of the Theory of Syntax (1965). This revised model is widely referred to as the Standard Theory of Transformational Grammar and is the version of Transformational Grammar described here.

Traditionally, research has focused on the structure of sentences as they appear, for instance, on paper. Grammar rules are used to specify the structure of each sentence. With Transformational Grammar, Chomsky proposed using grammars to exhibit the derivation process of human language: the knowledge and rules that humans use to create and understand utterances. Because it attempts to describe the knowledge and rules that 'generate' natural language, Transformational Grammar is a generative grammar and is often called Transformational Generative Grammar.

The derivation of a sentence, in Transformational Grammar, takes place in two phases. The initial phase uses a phrase structure grammar to generate a level of linguistic structure called the deep structure of the sentence. The deep structure is a representation of the full meaning of the sentence, with all of its intended nuances.

Consider the following sentences (example from [Winograd;1983]):

(19a) "Sam is easy to satisfy."

(19b) "Sam is eager to satisfy."

While these sentences have superficial similarities, a reader is immediately aware of the different relationships presented. In the first sentence, Sam is the person who can be satisfied. The one who acts to satisfy him is unidentified. In the second sentence, Sam is the person who does the `satisfying' and an unidentified person benefits. The syntactic structures of the surface manifestations of these sentences, as shown below, do not exhibit this difference.

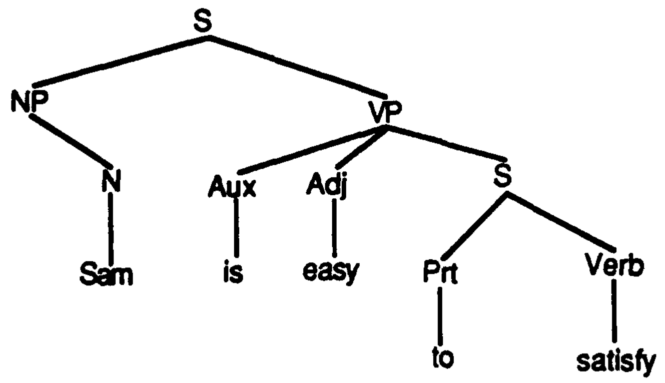


Figure 12
Surface Structure of "Sam is easy to satisfy."

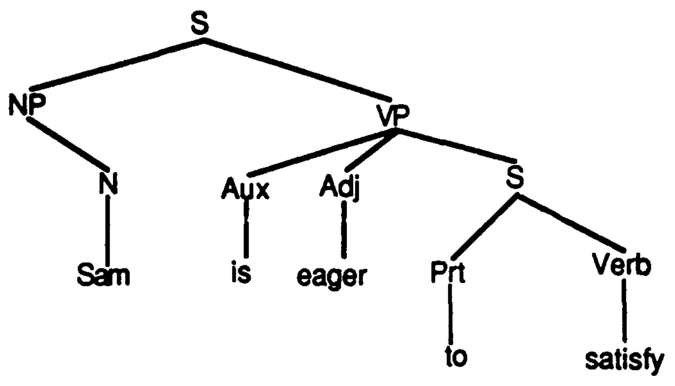


Figure 13
Surface Structure of "Sam is eager to satisfy."

The proposed deep structures for the sentences, shown in Figures 14 and 15, display the different meanings of the sentences.

Transformational Grammar postulates that two sentences with similar surface structures and differing implications may have differed significantly earlier in their derivation. Conversely, two sentences may have underlying similarities that are not apparent in their surface manifestations.

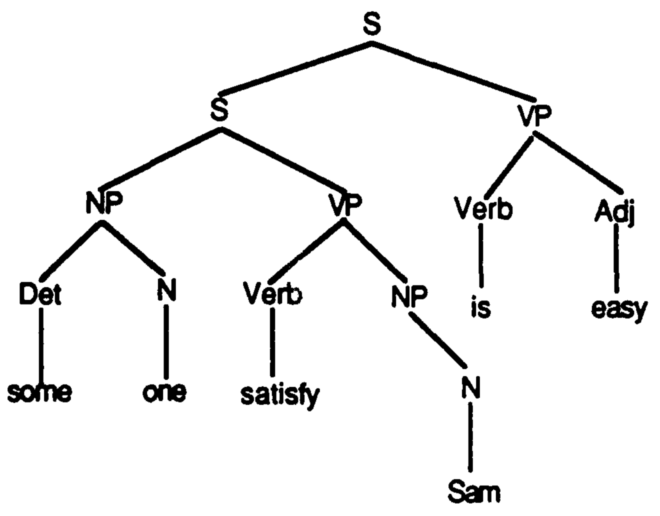


Figure 14
Deep Structure of "Sam is easy to satisfy."

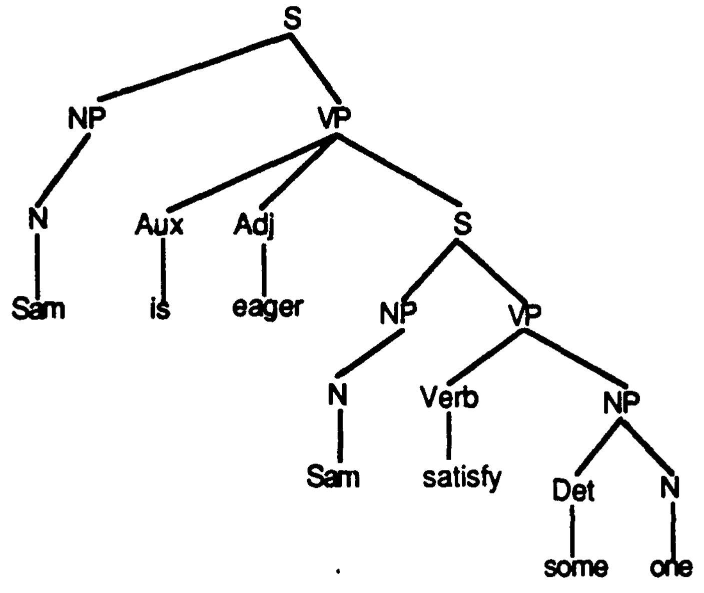


Figure 15
Deep Structure of "Sam is eager to satisfy."

In contrast to other phrase structure grammars, the rules in the first phase of Transformational Grammar rarely introduce words into the phrase structure. Rewrite rules such as "V -> shout" and "N -> boy" are eliminated. In their place are rules which form strings of complex symbols, such as [+Common] and [-Human], depicting syntactic "features" of the concepts they represent. These symbols act as selectional restrictions, placing restrictions on which words may fill positions in the phrase structure.

A feature with a plus (+) sign implies that a concept has that feature. A minus (-) sign implies that the concept is without that feature. Thus, the set of features [+N, +Common] implies a common noun such as "boy" or "book", whereas [+N, -Common] implies a proper noun such as "John" or "Egypt". [+Count] or [-Count] specifies a "countable" (for example, "dog") or "non-countable" (for example, "dirt") concept.

The following might be among the rules of a grammar [Chomsky; 1965]:

(20a)	N -> [+N, +/-Common]	(20d)	[-Common] -> [+/-Animate]
(20b)	[+Common] -> [+/-Count]	(20e)	[+Animate] -> [+/-Human]
(20c)	[+Count] -> [+/-Animate]	(20f)	[-Count] -> [+/-Abstract]

The first rule will rewrite the symbol N as either of the complex symbols

[+N, +Common] or [+N, -Common]. If N is rewritten as [+N, +Common], then the second rule states that either [+Count] or [-Count] is to be added to the set of complex symbols. This continues, building a set of complex symbols at each terminal node of the phrase structure.

To complete the phrase structure, each set of complex symbols is matched against the sets of complex symbols in the lexicon. Each item in the lexicon, or dictionary, of Transformational Grammar consists of a word and its description in terms of its syntactic features. When a set of complex symbols from the phrase structure matches a set from the lexicon, the associated word may be added to the phrase structure as the terminal node. Items in the lexicon of a Transformational Grammar might look like:

(21a) (sincerity, [+N, -Count, +Abstract])
(21b) (boy, [+N, +Count, +Common, +Animate, +Human])

In the second phase of Transformational Grammar special rules operate on the deep structure to produce the surface manifestation, called the surface structure, of the sentence. These special rules, called transformations, can each do one or more of the following elementary operations: they can move (reorder) phrase markers from one part of the structure to another, they can add copies of existing phrase markers to the structure, they can insert new structure, or they can delete structure. Transformations are applied one after the other in a strict order. Each consists of a pattern that must be matched by some cross-section of the phrase structure tree in order to be applicable, and a pattern defining the result of using the transformation. Some rules are obligatory (if they are applicable they must be run) and others are optionally applied. Thus, the specific surface structure derived from a deep structure is determined by whatever optional transformations are used. The structures that make up the deep structure of a sentence are sometimes referred to as the underlying phrase markers. When a transformation operates on the underlying phrase markers, the results are called derived phrase markers. These are transformed into other derived phrase markers, and so on, until all of the obligatory and the selected set of optional rules are run. The surface structure or final derived phrase marker results.

To illustrate the transformational process, observe the application of the optional Passive Transformation Rule as shown in the following.

(22) PASSIVE RULE:

$$\begin{tikzcd} \boxed{\begin{matrix} \text{NP - Aux - V - NP} \\ 1 \quad 2 \quad 3 \quad 4 \end{matrix}} \arrow[d] \\ \boxed{4 \quad 2 \quad \text{be} \quad \text{en} \quad 3 \quad \text{by} \quad 1} \end{tikzcd}$$

The pattern to match in order to apply this rule is NP - Aux - V - NP. The numbers are for easy referencing of the phrase markers. The Passive Transformation rule is applicable to the phrase structure of, "The Colts have beaten the Jets", shown in Figure 16.[2]

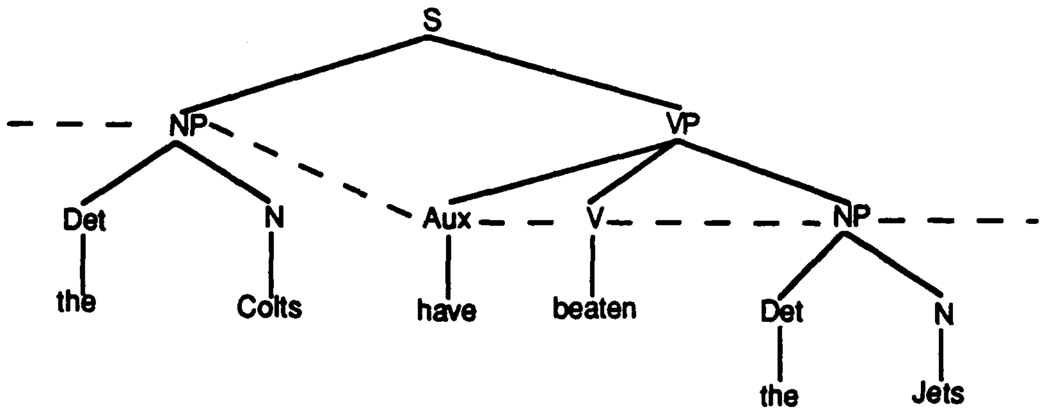


Figure 16
Syntactic Structure of"The Colts have beaten the Jets."

The phrases are matched as follows:

(23) the Colts have beaten the Jets
1 2 3 4

and transformed into:

(24) 4 2 be en 3 by 1
the Jets have been beaten by the Colts

2 The intermediate level of complex symbols is not shown.

A list of the names of the transformational rules and the ordering proposed for rule application is included as Appendix A to this report. In-depth explanations of each can be found in [Akmajian and Heny; 1975].

Transformational Grammar Theory attempts to describe human `generation' of Natural Language. Theoretically, inverse Transformational rules can reverse the process, transforming the surface structure of an utterance into its deep structure in order to "parse" the input.

While focusing its attention on the process as recounted here, Transformational Grammar Theory also makes a brief statement about how humans use the structures produced. The deep structure is proposed as the input to a human semantic component where semantic interpretation rules operate to derive a meaning representation. The surface structure is the proposed input to a phonological component where the pronunciation of the utterance is defined.

5. Modifications to the Standard Theory oı Transformational Grammar

Since the introduction of the Standard Theory of Transformational Grammar, a number of modifications and extensions have been proposed:

Extended Standard Theory (EST)

In the Standard Theory of Transformational Grammar, semantic interpretation rules operate on a single tree representation of the syntactic stucture of a sentence, the deep structure, to produce its semantic interpretation. Thus, as stated by the Katz-Postal Hypothesis (1964), transformations must be meaning-preserving. The application of transformations could not affect the semantic interpretation of a sentence. Soon after the introduction of the Standard Theory however, critics pointed out that meaning could be affected by the application of some transformations. As an example, the following pair of sentences, with obviously differing implications, would derive from the same deep structure:

(25) Not many arrows hit the target.
Many arrows didn't hit the target.

The Extended Standard Theory of Transformational Grammar proposed that semantic interpretation rules operate on the entire set of trees developed during the transformational process, rather than on the single, deep structure.

Generative Semantics

Proponents of Generative Semantics contend that the structure produced in the first phase of the Standard Theory, rather than being strictly a syntactic structure, is a semantic representation of the sentence. Transformational rules, as in the Standard Theory, operate to transform the base structure into its surface syntactic representation. The name `Generative Semantics' is derived from the viewpoint that language is generated from a semantic component.

The semantic base structure of Generative Semantics has a formal nature similar to a syntactic phrase structure representation in that it can be represented in tree form. The labels on its non-terminal nodes are some of the same categories as those used in syntactic tree structures (ex. S, NP,...), although presumed to be fewer in number. The terminal nodes of the base structure are semantically interpretable terms, similar to symbolic logic terms, instead of words. During the transformational process the terminal nodes may combine in a manner similar to symbolic logic processing.

Montague Grammar

The Montague Grammar formalism was devised by logician Richard Montague. The first phase of Montague Grammar uses a categorial grammar to construct the phrase structure representation of a sentence. A categorial grammar is a representation system for context-free grammars. A syntactic formula (rather than a single, syntactic class) is associated with each word in a categorial grammar dictionary. The grammaticality and syntactic structure of a sentence is determined by combining the formulas associated with each of its words using specified ordering and cancellation properties. For example, assume a noun, such as "gasoline", is expressed with the simple formula N and an intransitive verb, such as "explodes", is expressed as S/N. Then "gasoline explodes" is N x (S/N) or, by cancellation, S (a sentence). A transitive verb might have the formula (S/N)/N, implying it is something that combines with a noun (the `object' of a sentence) to produce an intransitive verb.

In Montague Grammar, rules representing semantic information are associated with each syntactic formula. In its second phase, the rules associated with each node of the syntactic tree tell how to combine semantically interpretable logic statements to form a single, formal logic representation of a sentence's meaning. The difficult notation and very formal nature of the combining mechanism (intensional logic) is quite intimidating. However the basic gist of Montague Grammar is evident in the following extremely simplified version of the process.

A noun phrase in Montague Grammar is represented by the set of properties of the entity which it describes. Thus, the noun phrase "Bill" is represented by the set of properties of the entity Bill. (Anything that is true of Bill is part of the description of what Bill is.) If the facts that Bill sleeps, talks, has blond hair and blue eyes, are represented as:

```
(26) (sleeps  Bill)
     (talks  Bill)
     (blond  Bill)
     (blue-eyed  Bill)
```

then this is the set (or subset) of properties that represent the noun phrase "Bill".

Intransitive verbs in the Montague formalism are represented by sets of entities. "Sleeps", for example, has as its meaning the set of entities that sleep:

```
(27) (sleeps  Bill)
     (sleeps Joe)
     (sleeps  Jill)
         .
         .
         .
```

In this example, the entity "Bill" is represented as the set of all predicates X such that (X Bill), and "sleeps" is represented as the set of all entities Y such that (sleeps Y). Now, the rule for combining the representation for the noun phrase "Bill" with that of the verb "sleeps" produces (sleeps Bill), or "Bill sleeps".

Montague Grammar is primarily concerned with assigning truth values to simple declarative sentences in models of the world. Hence, the final result of our simplified example is particularly fortuitous since the truth value of "Bill sleeps" can be determined by searching for its representation, "(sleeps Bill)", in the database representation of the state of the world.

Trace Theory

In Trace Theory, transformational rules are greatly reduced in number and complexity from those in the Standard Theory. The transformations that remain (there are only one or two) move structure from one position in the syntactic tree to another.

Surface structure representations of sentences in Trace Theory contain placeholders, called traces, that mark positions where structure was moved from at some time during the transformational process. A pointer from each trace indicates the moved item that it represents.

The surface structure of the sentence "What did John give to Sue", according to Trace Theory, is derived from the structure shown in Figure 17 ("John did give what to Sue"). A "Move" transformation moves "what" to the Complement position and leaves behind a trace (t), as shown in Figure 18.

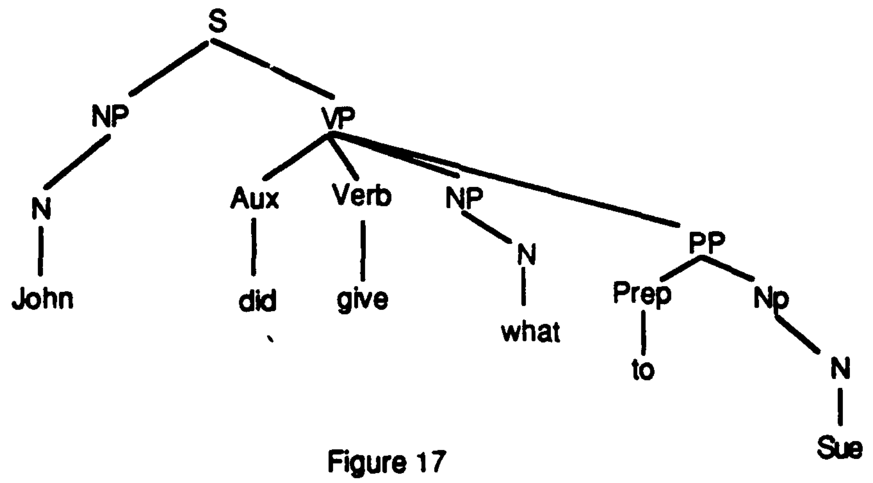


Figure 17
Syntactic Structure of "John did give what to Sue."

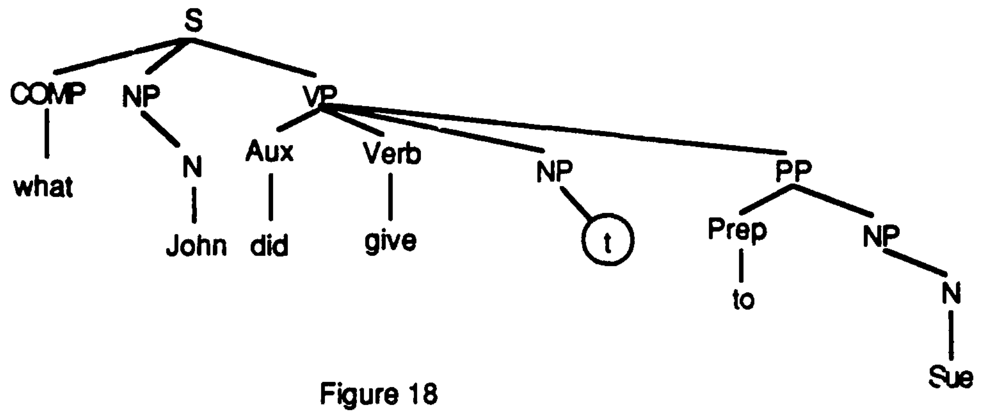


Figure 18
Syntactic Structure of "What did John give to Sue?"

Generalized Phrase Structure Grammar (GPSG)

Transformational Grammar was developed in response to what was widely seen as the inadequacy of phrase structure grammars for the description of natural language. However some researchers have recently presented counter-arguments to criticisms of the phrase structure grammar formalism. Most notably, Gerald Gazdar of the University of Sussex maintains that none of the arguments against phrase structure grammars are valid, and offers an extended interpretation of such grammars. This interpretation, called Generalized Phrase Structure Grammar (GPSG), includes the use of: complex symbols as in Transformational Grammar, rule schemata, and meta-rules. Rule schemata are patterns of rules. They present sets of rules, which have some common property, as a single statement. For example, the rule:

(28) * -> * "and" *, where * is any category

represents:

(29) NP -> NP "and" NP
VP -> VP "and" VP
N -> N "and" N
.
.
.

A metarule is a rule which creates new phrase structure rules from rules which already exist. If a grammar contains the following set of rules:

```
(30) VP -> V NP
     VP -> V NP NP
     VP -> V NP PP
     VP -> V NP VP
```

then the following metarule, where W represents any category:

```
(31) VP -> V NP W   =>   VP[PAS] -> V W PP
```

adds these rules to the grammar:

```
(32) VP[PAS] -> V PP
     VP[PAS] -> V NP PP
     VP[PAS] -> V PP PP
     VP[PAS] -> V VP PP
```

As in Montague Grammar, semantic rules are associated with each syntactic rule and operate in parallel to the syntactic analysis to create the semantic representation of a sentence.

Lexical-Functional Grammar (LFG)

The context-free grammar and the dictionary of a Lexical-Functional Grammar have an equation associated with each grammar rule and each dictionary entry. The first phase of an LFG generates a phrase structure tree from the grammar, with the leaf nodes of the tree selected from the dictionary. Next, the equations associated with each node of the phrase structure and each lexical entry are used to produce a representation of the sentence called the "functional structure".

The sample LFG rules shown in Figure 19 and the sample dictionary items displayed in Figure 20 are from [Winograd; 1983]. The parse of "A girl handed the baby the toys" would produce the phrase structure tree of Figure 21. Note that a unique variable is assigned to each `pre-lexical' tree node.

$$S \longrightarrow \begin{array}{c} NP \\ (\uparrow \text{Subject}) = \downarrow \end{array} \quad \begin{array}{c} VP \\ \uparrow = \downarrow \end{array}$$

$$NP \longrightarrow \text{Det} \quad \text{Noun}$$

$$VP \longrightarrow \begin{array}{c} \text{Verb} \\ \uparrow = \downarrow \end{array} \left(\begin{array}{c} NP \\ (\uparrow \text{Object}) = \downarrow \end{array} \quad \left(\begin{array}{c} NP \\ (\uparrow \text{Object2}) = \downarrow \end{array} \right) \right)$$

Figure 19

Sample Lexical-Functional Grammar (LFG) Rules

a	Det	(↑Definiteness) = Indefinite (↑Number) = Singular
baby	Noun	(↑Number) = Singular (↑Predicate) = 'Baby'
girl	Noun	(↑Number) = Singular (↑Predicate) = 'Girl'
handed	Verb	(↑Tense) = Past (↑Predicate) = 'Hand< (↑Subject), (↑Object), (↑Object2)>'
the	Det	(↑Definiteness) = Definite
toys	Noun	(↑Number) = Plural (↑Predicate) = 'Toy'

Figure 20
Sample Lexical-Functional Grammar (LFG) Lexicon

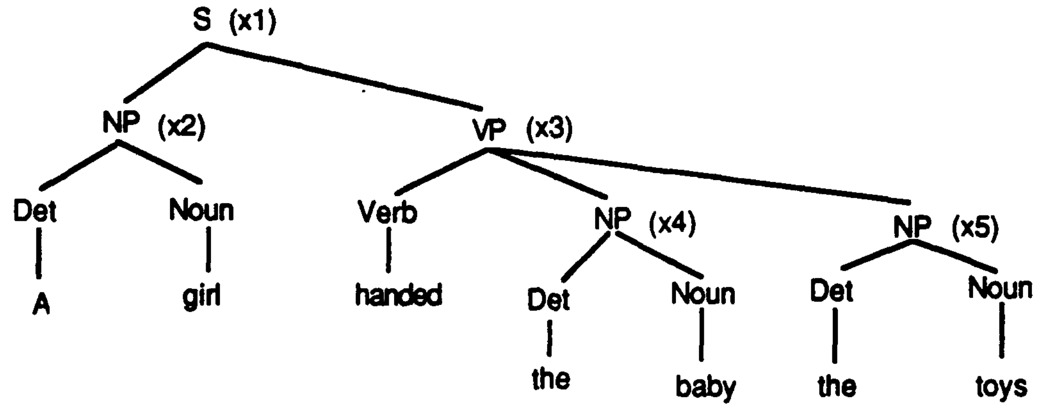


Figure 21
Syntactic Structure of "A girl handed the baby the toys."

In the second phase of the analysis process, equations associated with the rules which were used to derive the phrase structure tree are "instantiated": the up-arrows (↑) are replaced by the variable at the node where the grammar rule was applied, and each down-arrow (↓) is replaced by the variable at the node associated with the symbol under which the equation appears. With reference to our example, the equations associated with the topmost (S) node of the phrase structure tree derive from the grammar rule:

$$
(33) \quad S \longrightarrow \begin{array}{cc} NP & VP \\ (\uparrow \text{Subject}) = \downarrow & \uparrow = \downarrow \end{array}
$$

Here, the up-arrow is replaced by x1 (representing the node S of the phrase structure), the first down-arrow is replaced by x2 (representing the NP of the phrase structure), and the second down-arrow becomes x3 (the VP node). The instantiated equations for the S node then, are:

(34) (x1 Subject)=x2
and x1=x3

The equations associated with each lexical entry are somewhat similarly instantiated, however the up-arrow is replaced by the variable representing the parent of the parent node (two nodes up), and the down-arrow is replaced by the variable of the parent node of the lexical entry. (Thus, instantiated equations for `baby' are "(x4 Number) = Singular" and "(x4 Predicate) = `Baby'".) The system of equations derived from the second phase of analysis is then "solved", producing a single functional structure representing the sentence. The functional structure produced for "The girl handed the baby the toys" is shown in Figure 22. If there is no solution for the system of equations produced then the sentence is ungrammatical. More than one solution implies that the sentence is ambiguous. (For full details of this example see pages 334 - 338 of [Winograd; 1983].)

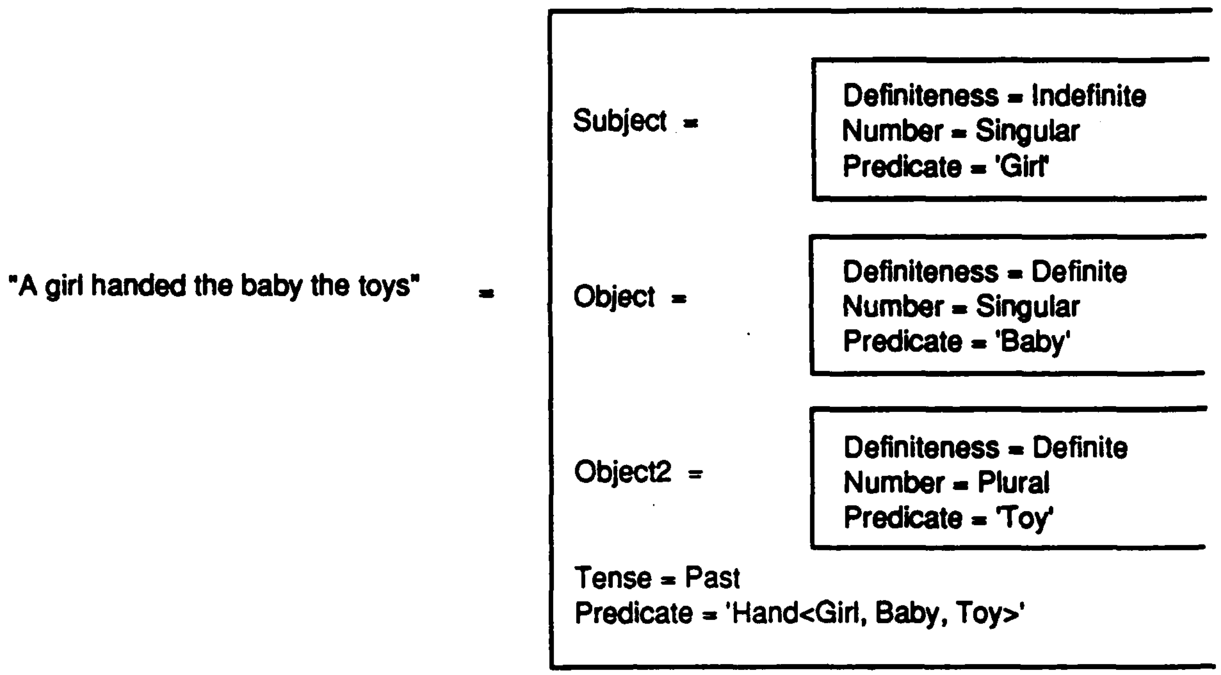


Figure 22
Functional Structure for "A girl handed the baby the toys."

Government-Binding (GB) Theory

Existence of the complicated system of rules of early Transformational Grammar Theory was difficult to justify in view of the speed with which children acquire language and the minimal breadth of experience they need before demonstrating language competence.

Government-Binding (GB) Theory, the modern grammatical theory most directly derived from early Transformational Theory, proposes the presence of only a single transformational rule called Move-α. Move-α allows any constituent to be moved anywhere. The result of an application of Move-α is then filtered through a number of "universal principles". The filtering mechanism rejects any ungrammatical utterance.

The universal principles are innate, not learned. People are thought to be naturally equipped with them. The exact number and description of the principles are not put forward by the theory. In that sense, GB Theory is still evolving. Some of the principles that have been defined, however, are the Case Theory, the Projection Principle, and the θ-Theory. In Case Theory, for example, noun phrases in most instances must be assigned a Case. (In many languages a noun's Case is marked by a variation in the form of the word. In English, only personal pronouns demonstrate Case distinctions. For example, "they" is used as the Subject of a sentence and "them" is used as an Object.) If a noun phrase does not have a Case assigned to it, or fails to be in a position designated by its Case, then the structure is ruled ungrammatical.

The universal principles of GB Theory are general constraints. Chomsky demonstrated, in [Chomsky; 1977], that many superficially dissimilar rules from early Transformational Grammar Theory share many important properties. Thus, the universal principles of GB Theory combine to produce effects that previously were produced by a single transformation.

Proponents of GB Theory note that some human languages have similar - but different - constraints. For example, there may be two languages in which a noun phrase cannot be moved from a certain structural position, but the specific structural position may be different for each language. GB Theory uses "parameters" to maintain the universality of its principles. For the example given, the appropriate single constraint may specify that 'a constituent cannot be moved from position X, where X is a variable over a small number of values, X_1, X_2,..., X_N. The setting of all of the parameters in the constraints identifies the particular human language; set them one way to get grammatical French, another way to get English.

[Sells; 1985] and [Berwick and Weinberg; 1986] provide good discussions of GB Theory and much more thorough descriptions of the currently accepted array of principles.

B. Tools and Techniques for Implementing Syntactic Analyzers

Various techniques for representing and maintaining the details of the syntactic analysis of sentences are in current use. A number of them are described here:

Augmented Transition Networks (ATNs)

An Augmented Transition Network (ATN) is a formalism for representing syntactic grammars. An ATN consists of a set of nodes, representing states of the parse, and directed arcs which connect the nodes into networks. These networks of nodes are "augmented" with conditions and actions attached to the arcs, and "registers" which may be used to store intermediate results of the parse. The conditions (or tests) associated with each arc determine whether or not a transition may be made to the next node. Often, transitioning through an arc requires that a subnetwork be satisfied. Actions associated with arcs are executed when transitions are made through the arcs. An input sequence to an ATN is deemed to be a sentence if, beginning at the (unique) start state, an analysis of the input leads to a (not necessarily unique) final state of the network.

As stated in section III of this report, there are often occasions during the parsing process when two or more grammar rules are applicable. Similarly, ATNs often have two or more paths away from a node. Most implementations of ATN parsers can select a path with the option of later backtracking and selecting a different path should the first prove unsuccessful.

Following part of a parse will demonstrate the ATN concept. A small dictionary, a network with associated Conditions and Actions, and a set of Registers are displayed in Figures 23, 24, and 25. Note that any arc labeled "JUMP" is automatically satisfied. For the sake of brevity and clarity assume that the correct choice is made at each instance in which there are multiple possible paths.

Word	Category	Features
a	Det	Number: Singular
black	Adj	
cat	Noun	Number: Singular
chased	Verb	Transitivity: Transitive Tense: Past
fish	Noun	Number: Singular or Plural
	Verb	Transitivity: Intransitive
girls	Noun	Number: Plural
mouse	Noun	Number: Singular
saw	Noun	Number: Singular
	Verb	Transitivity: Transitive
the	Det	
These	Det	Number: Plural

Figure 23

Sample ATN Dictionary Entries

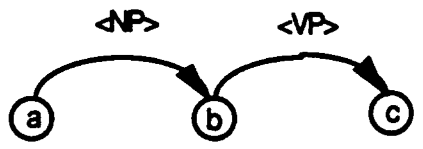
Subject:	________
Number:	________
Main-Verb:	________
Auxiliaries:	________
Direct-Object:	________

Figure 24

Sample ATN Registers

Arc-associated Actions and Conditions:

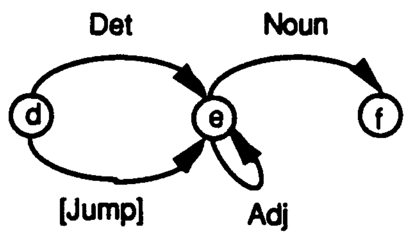
S:



<u>a-NP-b</u> (From node "a" to node "b")

Action: Set Subject to current constituent.

NP:



<u>d-Det-e</u>

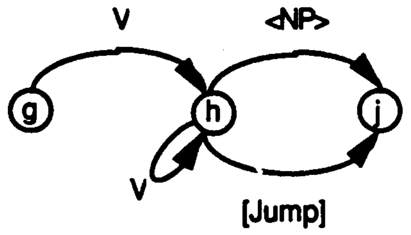
Action: Set Number to the Number of the current constituent.

<u>e-Noun-f</u>

Condition: Number is empty or Number matches the Number of the current constituent.

Action: Set Number to the Number of the current constituent.

VP:



<u>g-V-h</u>

Action: Set Main-Verb to current constituent.

<u>h-V-h</u>

Condition: The Main-Verb is form of Be, Do, or Have.

Action: Add contents of Main-Verb to Auxiliaries.
Set Main-Verb to current constituent.

<u>h-NP-j</u>

Action: Set Direct-Object to current constituent.

Figure 25
Sample Augmented Transition Network

Start at node "a". Assume the input sentence is, "The black cat chased a mouse".

The S Network: The only route away from node "a" connects to node "b" and has no associated pre-Conditions to satisfy. Crossing the arc requires the satisfaction of the NP subnetwork, i.e. finding a noun phrase at the beginning of the sentence. Consider the S network to be temporarily "on hold" and move to node "d" of the NP network.

The NP Network: There are two paths away from node "d". One, the "JUMP" arc, takes us directly to node "e" with no pre-Conditions, no follow-up Actions, and, importantly, no matching of input. The other path has no pre-Conditions but requires that the first word of the input sentence be a determiner. After following the latter of the two paths to node "e" the remaining input is "black cat chased a mouse". The Action associated with the just-crossed path says to set the Number register to be the same as the Number feature associated with "the current constituent". The dictionary does not associate a Number feature with the word "the", so the Number register remains empty. At node "e" there are again two possible paths. Follow the path from "e" back to "e". There were no pre-Conditions for this arc. The adjective "black" has been matched and the input remaining is, "cat chased a mouse". At the present state of the parse we are at node "e" and, as before, there are two paths leading away. The pre-Condition on the arc to node "f" requires that the Number register be empty (which it is) or that the contents of the Number register match the Number feature of the current constituent "cat". This pre-Condition is satisfied, follow the path to node "f". The Action required after crossing to node "f" sets the Number register to "Singular", the Number feature for "cat". Now the NP subnetwork has been satisfied.

Return to the S Network: In the S network, we are now allowed to move from node "a" to node "b". The Action associated with this move sets the Subject register to the noun phrase "the black cat". The single path leading from node "b" requires that the VP subnetwork be satisfied, so, as before, the S network will be put "on hold".

The parsing process would proceed by transitioning through the VP network, then returning to toplevel, the S network, where the parse is considered successful upon reaching node "c".

James Thorne, Paul Bratley, and Hamish Dewar [Thorne, et al; 1968] are credited with the development of the ATN mechanism. Its use was demonstrated and popularized by William Woods' LUNAR system.

Chart Parsing

Charts are used to keep track of the syntactic parse in progress. Charts record syntactic constituents found so far.

A chart is a network of vertices linked by edges. Each edge represents a constituent part of the input; its vertices define the beginning and the end of the constituent. A chart parse begins with a structure containing only the individual words of an input as its edges. Thus, the initial chart formed for the parse of "The black cat bit the dog" is shown in Figure 26, below.

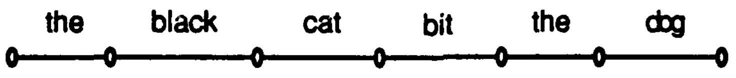


Figure 26

Next, edges labeled with the lexical category of each word are added. (See Figure 27. Note that "bit" can be either a noun or a verb.)

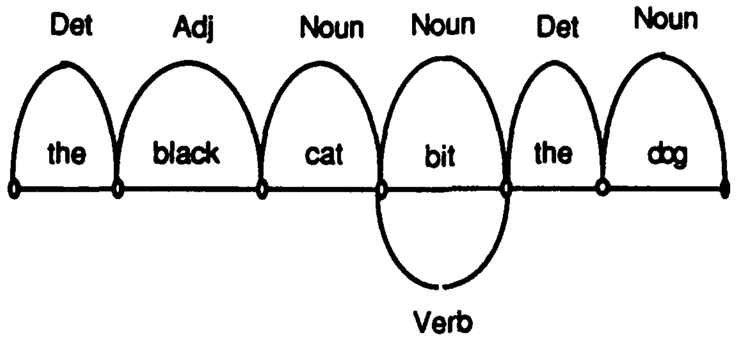


Figure 27

During the parse, a new edge is added to the chart whenever a new constituent is found. If the rules "NP -> Det Noun", "NP -> Det Adj Noun", and "NP -> Det Adj Noun Noun" are included in the grammar, the chart structure shown in Figure 28 will be constructed for our example.

A successful parse of a sentence is found when an edge extends from the first vertice to the last. More than one such edge indicates ambiguous input.

The chart constructed for "the black cat bit the dog" is shown in Figure 29. This analysis indicates the syntactic structure of the sentence to be as shown in Figure 30.

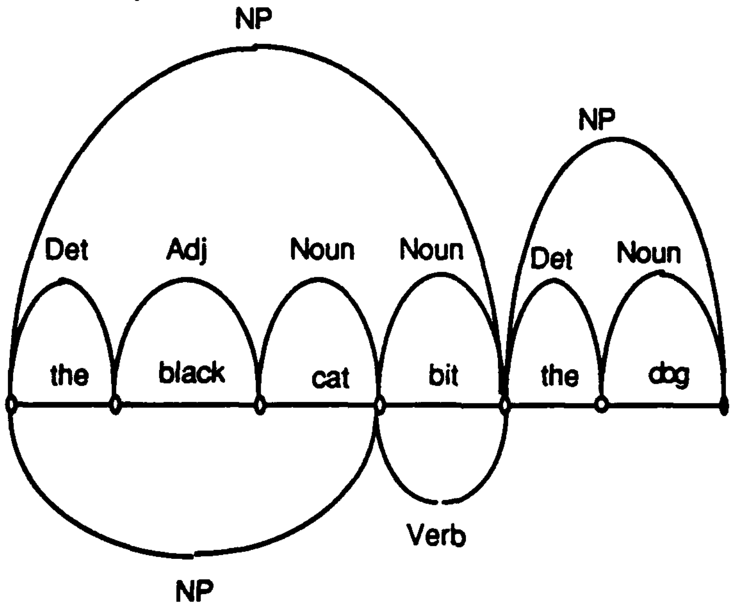


Figure 28

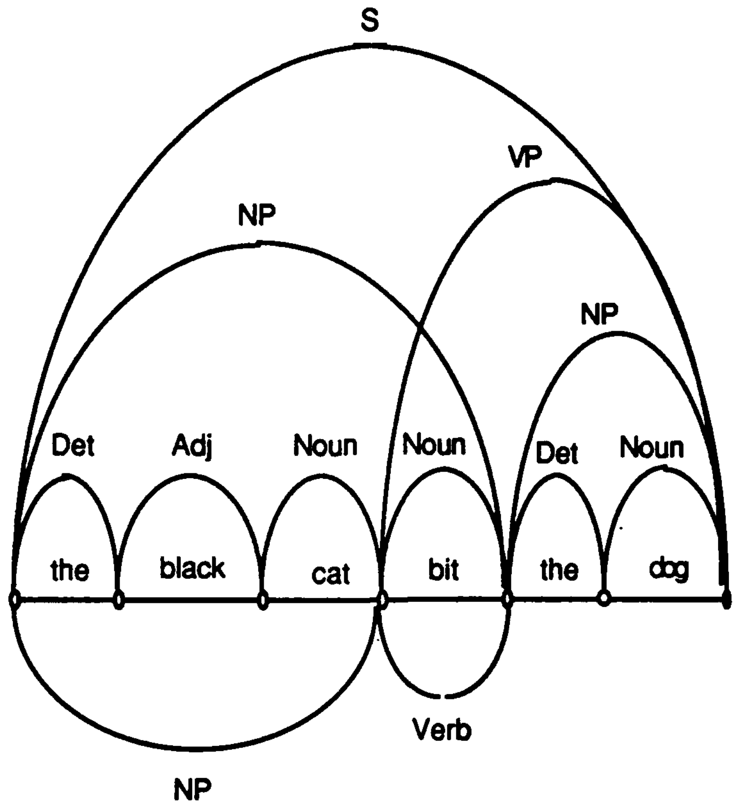


Figure 29

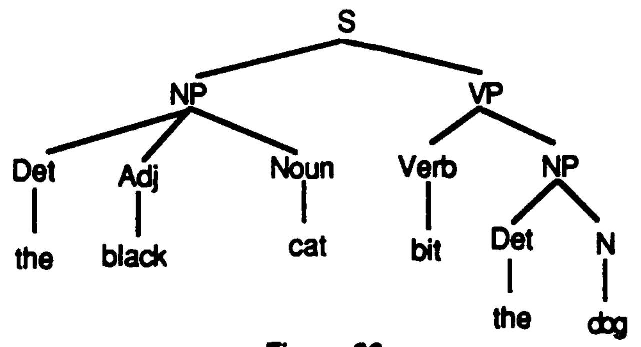


Figure 30

Chart parsing, as described above, only keeps track of grammar rules that succeed. Active Chart Parsers, on the other hand, keep track not only of rules that succeed but of all rules that have been tried. Active chart parsers have active edges and inactive (or complete) edges. Inactive edges represent whole constituents and indicate that a grammar rule has succeeded. An active edge represents the start of an incomplete constituent and would be represented by an arc that is anchored to the base line at only one end. The label on an active edge includes its "remainder", that is, the parts that need to be found in order to complete an edge.

The earliest use of chart parsing in a practical system was in the MIND system [Kay; 1973] for machine translation. MCHART [Thompson; 1983] is a system for constructing chart parsing systems.

Definite Clause Grammars (DCGs)

Context-free grammar rules may be expressed in the form of logic statements called definite clauses. The resulting grammar is called a Definite Clause Grammar (DCG). DCGs are often implemented computationally in the logic programming language Prolog. In fact, most implementations of Prolog include a grammar rule interpreter which automatically translates a grammar rule into executable code. Thus,

(35) "sentence --> noun_phrase, verb_phrase."

will be automatically translated to:

(36) "sentence(S0,S) :- noun_phrase(S0,S1), verb_phrase(S1,S)."

Appendix C contains a sample, very small, DCG grammar from [Clocksin and Mellish;1981]. CHAT-80, a prototype natural language question-answering system based on the DCG formalism, is described in [Warren and Pereira;1982].

V. Other Processing Techniques

"Other" processing techniques are those in which the syntax of the input is not the main focus of the analysis. The simplest such technique uses a keyword and simple pattern matching scheme. The method and a very well-known implementation of it are described below:

A. Keyword/Simple Pattern Matching Systems

Some functionality, although limited, is offered by systems which do simple matching of patterns against inputs. The particular pattern which is matched determines the response that will be given. No real analysis of input is done by these simple systems. A well-known example of such a system, called ELIZA, was developed by Joseph Weizenbaum [Weizenbaum; 1966].

In dialog with a human, ELIZA simulates a Rogerian psychotherapist. Responses from the system are meant to appear to be leading the patient (the person typing keyboard inputs to the system) towards self-discovery of truths about themselves. A typical conversation is shown below:

(37) PATIENT: Men are all alike.
ELIZA: In what way

PATIENT: They're always bugging us about something or other.
ELIZA: Can you think of a specific example?

PATIENT: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here.

PATIENT: He says I'm depressed much of the time.
ELIZA: I am sorry to hear you are depressed

ELIZA inspects each input sentence for any of several keywords and attempts to match the input against a pattern, called a decomposition rule, containing that keyword. Thus, if the patterns associated with the keyword "remember" are "(* I remember *)" and "(* do you remember *)", where asterisks represent a sequence of zero or more words, the input "I remember when winters were harsher" would match the first pattern. The second asterisk in the pattern would match the string "when winters were harsher". (The first asterisk is matched with zero of the input words.)

Each decomposition rule has an arbitrary number of reassembly rules associated with it which describe the format for the response. Possible reassembly rules for "(* I remember *)" include: "(Do you often think of 4)", "(Does thinking of 4 bring anything else to mind)", and "(What else do you remember)". Numbers in a reassembly pattern are an index to an item obtained from the match of the input to the decomposition rule. Here, the response would be "Do you often think of when winters were harsher", "Does thinking of when winters were harsher bring anything else to mind", or simply "What else do you remember".

A facsimile of Weizenbaum's program can be produced from the details provided in [Weizenbaum; 1966]. Another early system based on pattern matching, called SIR, is described in [Raphael; 1968].

B. Semantic Grammars

Semantic Grammars are grammars that use word classes specific to the domain being discussed rather than (or in combination with) the traditional grammar classes of Noun, Verb, Noun Phrase, and so on. For instance, a system dealing with airline reservations might have as classes: Destination, Flight Number, Flight Time, etc. A semantic grammar rule for such a system might be: S -> <PLANE-ID> is <PLANE-TYPE>.

The same tools used for syntactic analysis of input, ATNs and the like (described in IV.B.), can be applied when the grammar is a semantic grammar. For example, SOPHIE, described briefly in Section II of this report, uses an ATN and a grammar based on semantic categories such as Measurement, Transistor, and Circuit Element. An ATN used with a semantic grammar is called a semantic ATN.

C. 'Case'-Based Semantic Processing

In the traditional notion of "case", different forms of a word can provide information on the role played in the sentence by the concept represented. English, for example, has a case classification system for pronouns. Cases are: subjective, objective, reflexive, possessive determiner, and possessive. Hence, when referring to oneself in the subjective case, one would use the word "I", as in: "I went to the store". When referring to oneself objectively, one would use "me": "Joan went with me to the store". (Reflexive, possessive determiner, and possessive forms for the First Person Singular are, respectively: myself, my, and mine.) These so-called syntactic (or surface) cases vary from language to language and convey only minimal semantic information.

In semantic case grammar theories, an analysis of an input sentence produces a representation of its meaning (semantics) rather than its syntactic structure. Each such theory proposes the existence of a small number of universal, primitive, semantic (or deep) cases which are descriptive of the possible semantic roles of a noun phrase in a sentence. The following sentences demonstrate the idea of semantic cases:

(38) We baked every Wednesday evening.
The pecan pie baked to a golden brown.

"We" in the first sentence and "the pecan pie" in the second, hold the same syntactic position (the initial noun phrase in a syntactic parse), but each plays a different semantic role. "We" describes WHO performed the action of baking; "the pecan pie" tells WHAT object the action of baking was performed on.

The first notable presentation of semantic case grammars was by Charles Fillmore in a paper entitled "The Case for Case" [Fillmore,1968]. The primitive semantic roles proposed by Fillmore were:

(39) Agentive (A) The instigator of the action, usually animate.
For example, John in "John opened the door".

Instrumental (I) The inanimate force or object involved in the action.
The key is the Instrumental in "John opened the door with the key" and "The key opened the door".

Dative (D) The animate being that is affected by the action.
Thus, John is the Dative in "We persuaded John that he would win".

Factive (F) The object that results from the action.
The Factive in "We made John a jacket" is the jacket. The dream is the Factive in "John had a dream about Mary".

Locative (L) The location or spatial orientation of the action.
Chicago is the Locative in "It is windy in Chicago".

Objective (O) The object affected by the action.
For example, the door in "The key opened the door" and "John opened the door with the key".

According to Fillmore's Theory, each verb sense has an associated case frame describing the set of cases that appear or are optional when that verb sense is used in a sentence. For instance, the case frame for the usual sense of the word "open" would dictate that the Objective case is obligatory, but the Agentive and Instrumental cases are optional. A representation for this case frame might be: [O, (A), (I)]. Parentheses indicate optional cases. Thus, a sentence whose main verb is "open" requires a filler for the Objective case (something which is `opened'), and may or may not have fillers for the Agentive (someone who does the `opening') and Instrumental (an object with which to do the `opening') cases. The ungrammaticality of "John opened with the key" is evidence that a filler for the Objective case is required for the word "open".

Yorick Wilks' ([Wilks; 1975]) English-to-French language translation system has a case grammar component. In the system, word senses are represented by "formulas" which can be expressed graphically with binary trees. The formula for "drink" (the action of "drinking" a liquid) is: "((*ANI SUBJ) (((FLOW STUFF) OBJE) ((SELF IN) (((*ANI (THRU PART)) TO) (BE CAUSE)))))". The tree representation is shown in Figure 33. According to the formula, the "preferred" (but not required) agent doing the drinking is animate, the object of the drinking is preferrably liquid (something that flows), the container into which the aforementioned liquid goes is the animate agent, and the direction of the action is to an aperture in the animate agent (the mouth). (In Wilks' system, verbs are not necessarily the only word senses with case preferences.) The system attempts to form a semantic representation for a sentence by meshing word sense formulas into a larger graphical structure. If more than one possible representation is formed, the one with the highest "semantic density" (the one in which the largest number of case preferences are satisfied) is assumed to be the correct representation. Acceptance of less than a perfect fit in this manner allows the system to understand non-standard word usage, as in "My car drinks gas". Wilks calls this technique for building a semantic representation Preference Semantics.

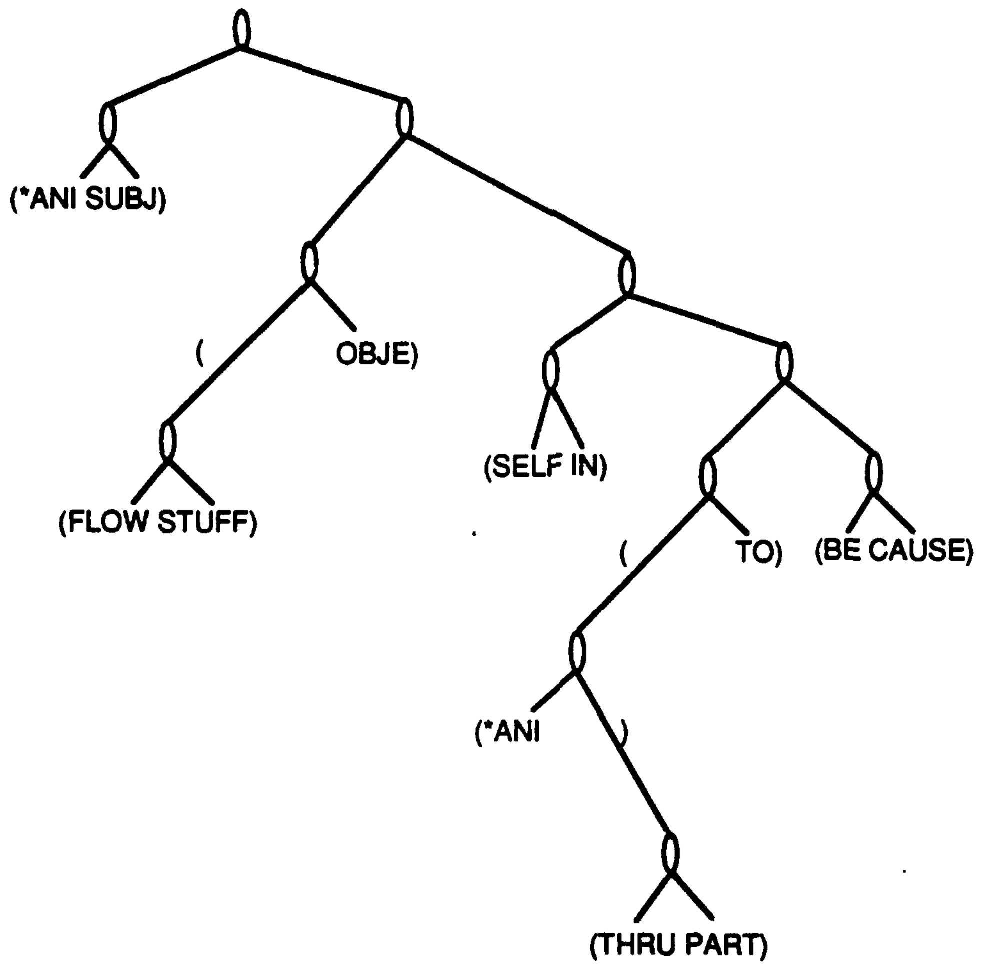


Figure 31
Tree Representation of Wilks' Formula for "drink"

Case grammar structures are present in most implementations of "semantic networks" and in turn, since it has derived from the semantic net formalism, are present in the "Conceptual Dependency" knowledge representation (see the following sections on "Semantic Networks" and "Conceptual Dependency Theory").

Semantic Networks

A semantic network is a knowledge representation consisting of "nodes" and labelled, directed "arcs" (arrows). Nodes in a semantic net represent entities or concepts. The arcs (sometimes called associative links or, simply, links) represent relationships among nodes by connecting them into a network. For example, Figure 32 is a very simple semantic network made up of two nodes (representing a person named Anna and a person named Bill), and an arc indicating "Bill likes Anna". Note that "Anna likes Bill" is not a relationship which can be assumed from this network.

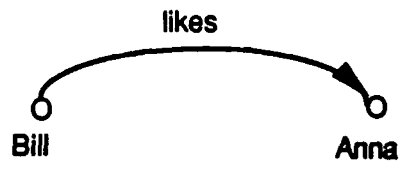


Figure 32
Semantic Network Representation of "Bill likes Anna" [Scragg; 1976]

A slightly more complicated relationship is represented by the network in Figure 33. Here, "Charles is a boy" and "Charles hit the tall girl" are represented. Notice that the node for "the tall girl" is an unlabelled concept. (In order to reference "the tall girl" this node could be labelled with an arbitrary, unique symbol.) Networks can be built up to represent very complicated sets of relationships.

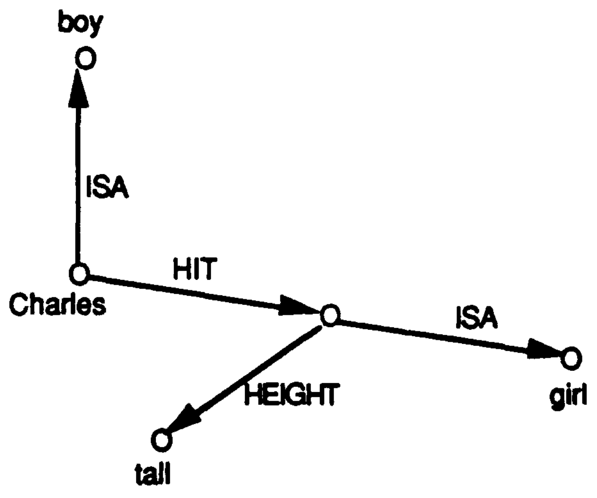


Figure 33
Sample Semantic Network

The basics of the semantic net formalism have been described. Additional functionality can be provided by organizing related concepts in the network from the most general to the most specific. For example, the net in Figure 34 shows that: a dog is an animal, a Schnauzer is a type of dog, and Bert is a Schnauzer. This organization of concepts builds an inheritance hierarchy: anything that is true about a general concept, unless otherwise noted specifically, is true of the concept below it on the hierarchy. Thus, since a dog is an animal and a Schnauzer is a dog, a Schnauzer is an animal (and Bert, because he is a Schnauzer, is a dog, and therefore is an animal, etc).

A further refinement of the semantic net formalism allows links to represent properties of concepts. For instance, in the network of Figure 34, NUMBER-OF-LEGS and VOCALIZATION characteristics are included as properties of dogs.

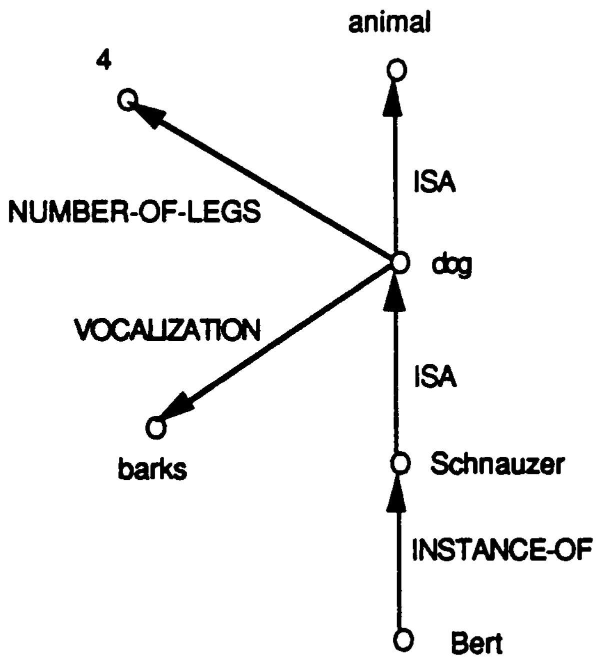


Figure 34
Another Sample Semantic Network

Inheritance of the concept properties (property inheritance) is enabled by the hierarchical organization of the network. To determine how many legs Bert has, one would look first at the Bert-node. If the NUMBER-OF-LEGS property is not present (or is present but no value is specified), the arc to the next, more general node would be traced. Since the target property is not found on the node representing "Schnauzer", one would look to the "dog" node. Here, NUMBER-OF-LEGS is given as 4. Thus Schnauzers, by property inheritance, have 4 legs and, (again by property inheritance) so does Bert. Properties inherited from more general classes act as default values. A value for a property may be specified at any level in the hierarchy and overrides any value for that property provided at a node which is higher in the hierarchy. Thus, if Bert happens to be a 3-legged dog, that fact can be recorded by specifying NUMBER-OF-LEGS to be 3 at the Bert node.

Notice that an INSTANCE-OF arc in Figure 34 indicates that Bert is a "specific instance" of Schnauzer. Were it not for this distinction, Bert could be mistaken for an empty subclass of Schnauzers. Some types of incorrect inferences are also avoided by this differentiation between classes and instances of things. If "Dogcatchers chase dogs" is true, then "Dogcatchers chase Schnauzers" is true, in general. However, it may not be the case that "Dogcatchers chase Bert". Nodes representing classes of objects are called type nodes. A node representing an individual is a token node.

Information is obtained from semantic networks most often by matching network patterns. For example, assume the existence of the semantic network of knowledge shown in Figure 35 (slightly altered version from [Barr and Feigenbaum; 1981], page 187).

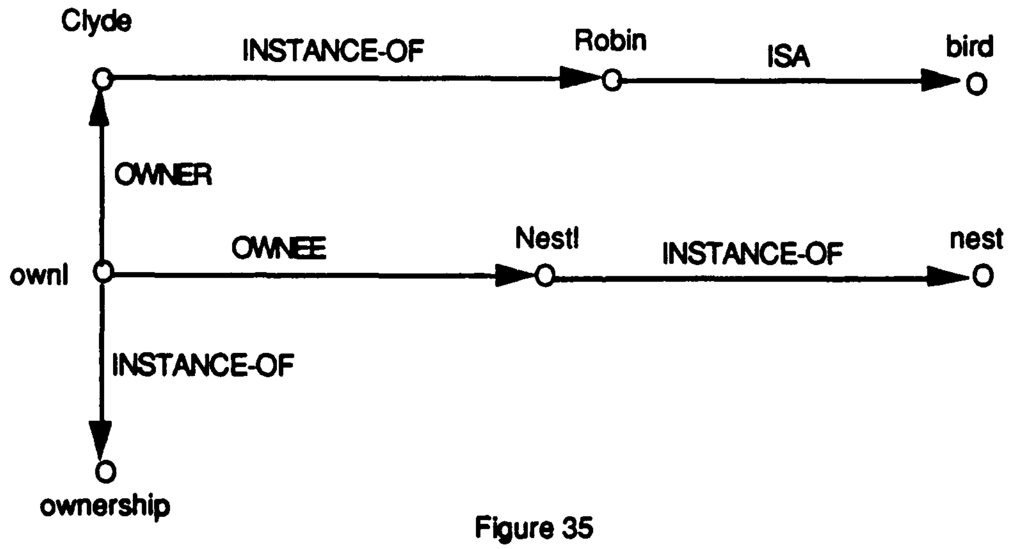


Figure 35
Semantic Network [Barr and Feigenbaum; 1981]

In order to answer the question "What does Clyde own?", the semantic fragment shown in Figure 36 would be constructed and matched against the database network providing the correct replacement for the variable X: Nest1.

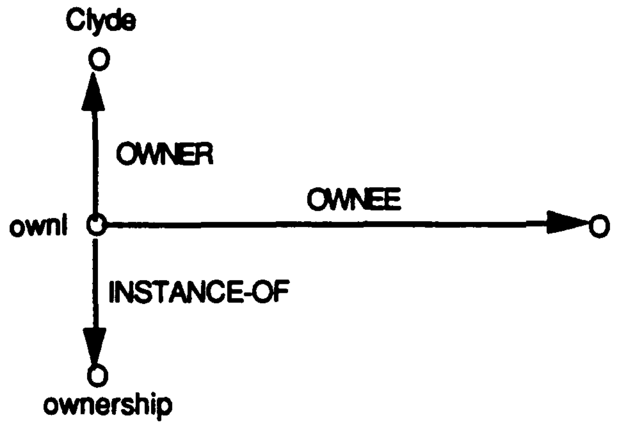


Figure 36
Semantic Network Repesentation of "What does Clyde own?"

Note from Figure 35 that nodes can represent more than just concrete objects, as in the "ownership" node representing a kind of relationship and the "own1" node which represents a specific instance of ownership. These concepts may be extended by associating a "case frame" with their nodes. Such a case frame would consist of a set of outgoing arcs to descriptors of the parent node. The case frame associated with the ownership node, for example, might consist of arcs labelled "OWNER", "OWNEE", "START-TIME", and "END-TIME". This case frame could be inherited by instances of "ownership", such as "own-1", where values for these properties could be assigned or left to default to a value assigned in the "ownership" node.

The most well-known, and perhaps the first, implementation of a semantic network was Quillian's Semantic Memory ([Quillian; 1968]). Proposed by Quillian as a model of human associative memory, the program accepted two words as input and attempted to return information on their semantic similarities or differences. Beginning from the nodes representing the two input words, "tags" were placed at each node that could be reached by following arcs until a path between the two words were found. At the conclusion of this spreading activation process, the path between the words was described in the format shown for input words "cry" and "comfort" in (40), below. (Selected senses of words in the intersection path are indicated by a numerical suffix, as in CRY2 meaning the second sense of "cry".)

(40) Input words: CRY, COMFORT

Intersection: SAD

(1) CRY2 is among other things to make a SAD sound.

(2) To COMFORT3 can be to MAKE2 something LESS2 SAD.

Quillian developed the Teachable Language Comprehender (TLC) [Quillian;1969] around the concept of a semantic memory. TLC "learns" to understand English text.

Conceptual Dependency, described next, is derived from the semantic network formalism.

Conceptual Dependency (CD) Knowledge Representation Theory

The basic axiom of Conceptual Dependency (CD) Theory states: "For any two sentences that are identical in meaning, regardless of language, there should be only one representation" [Schank and Abelson; 1977]. Proposed by Roger Schank as a formalism for representing meaning, CD theory attempts to represent the "conceptualizations" underlying sentences, rather than the sentences themselves. A small number of primitive ACTs, which are the basic meaning units for representing actions, and a number of primitive STATEs are used to construct representations of actions in an unambiguous manner. CD Theory proposed the existence of 11 primitive ACTs: ATRANS, PTRANS, MTRANS, MBUILD, PROPEL, MOVE, INGEST, EXPEL, GRASP, SPEAK, and ATTEND. Brief descriptions and examples of the types of events represented by each are provided in Appendix D.

Each primitive ACT has an associated "case frame" which describes the roles that are required for a full conceptualization of that primitive. Case frames provide information on the concepts that can be expected to appear in a sentence. These expectations help direct the parse towards a successful conclusion (expectation-driven processing).

All of the roles that are present in the case frame must be filled, none are optional. The ACTOR case is always present. Its filler tells who the performer of the action is. Other possible cases (or roles) are: OBJECT, RECIPIENT, DIRECTION, and INSTRUMENT. Since CD structures represent underlying conceptualizations, the concepts that fill the CD roles may or may not be present in their manifestations as sentences.

The representation of "John goes to New York" demonstrates the basics of CD theory as presented so far. "Going" is the transfer of the physical location of an object and, hence, is represented by a PTRANS. The ACTOR, or performer of the PTRANS, is John. The DIRECTION of the act is FROM some unknown place TO New York, and the OBJECT of the transfer of location is John. Thus, the CD representation for "John goes to New York" looks like (and is) the same as the representation for "John takes himself to New York" (see (41)).

(41) The CD representation of "John goes to New York".
[Schank and Riesbeck; 1981]

```
(PTRANS
        (ACTOR John)
        (OBJECT John)
        (DIRECTION  (FROM unknown)
                    (TO New York)))
```

In an actual CD implementation, role fillers in each of the examples of this section would be structures representing the concepts "John", "New York", and so on, rather than English descriptions as shown here.

Relationships among concepts are called dependencies. Thus, there are object dependencies, direction dependencies, etc. Progressively complex structures are formed when concepts depend on other concepts, as in the CD representation of "John saw Mary" (42).

The act of "seeing" is to transfer information from one's eyes to one's conciousness, and is done by directing one's eyes at whatever is to be seen. Therefore, there is an instrument dependency on an act of ATTENDing.

Note that "Mary's image", not Mary, is transferred from John's eyes. Also note, the place to which the image is transferred is represented as the mental location (MLOC) of John's Conscious Processor (CP).

Primitive STATEs provide information about some property or condition of objects. Many of these states can be described (for computational purposes especially) by a numerical value. For instance, the "HEALTH", "ANGER" level, and "CONSCIOUSNESS" states of an object may be valued as being in the range from -10 to +10. [Schank; 1975] gives the scale of values shown in (43) for describing an object's HEALTH.

(42) CD representation of "John saw Mary".
[Schank and Riesbeck; 1981]

```
(MTRANS
          (ACTOR John)
          (OBJECT image of Mary)
          (DIRECTION     (FROM John's eyes)
                         (TO MLOC  (CP of John)))
          (INSTRUMENT    (ATTEND
                                   (ACTOR John)
                                   (OBJECT eyes)
                                   (DIRECTION    (FROM unknown)
                                                 (TO   Mary)))))
```

(43) Sample scale values for HEALTH STATE of objects in CD.
[Schank; 1975]

HEALTH STATE (Scaled from -10 to +10):

dead	-10
gravely ill	-9
sick	-9 to -1
under the weather	-2
all right	0
tip top	+7
perfect health	+10

Other STATES, such as "LENGTH" and "COLOR", have absolute values ("45 feet" or "red", for example) rather than a scaled rating.

Early descriptions of the Conceptual Dependency knowledge representation schema used labelled arrows to graphically represent dependencies. The graphical notations for "John went to New York", "John saw Mary", and "John died" are shown in Figure 37, Figure 38, and Figure 39, respectively.

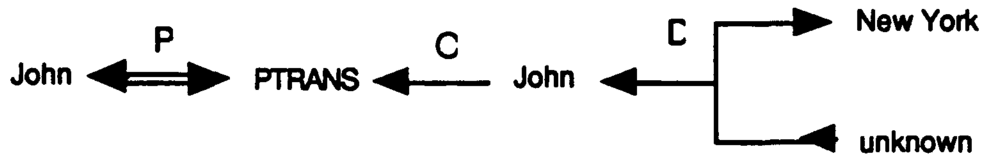


Figure 37
Graphical CD representation of "John went to New York." [Schank; 1975]

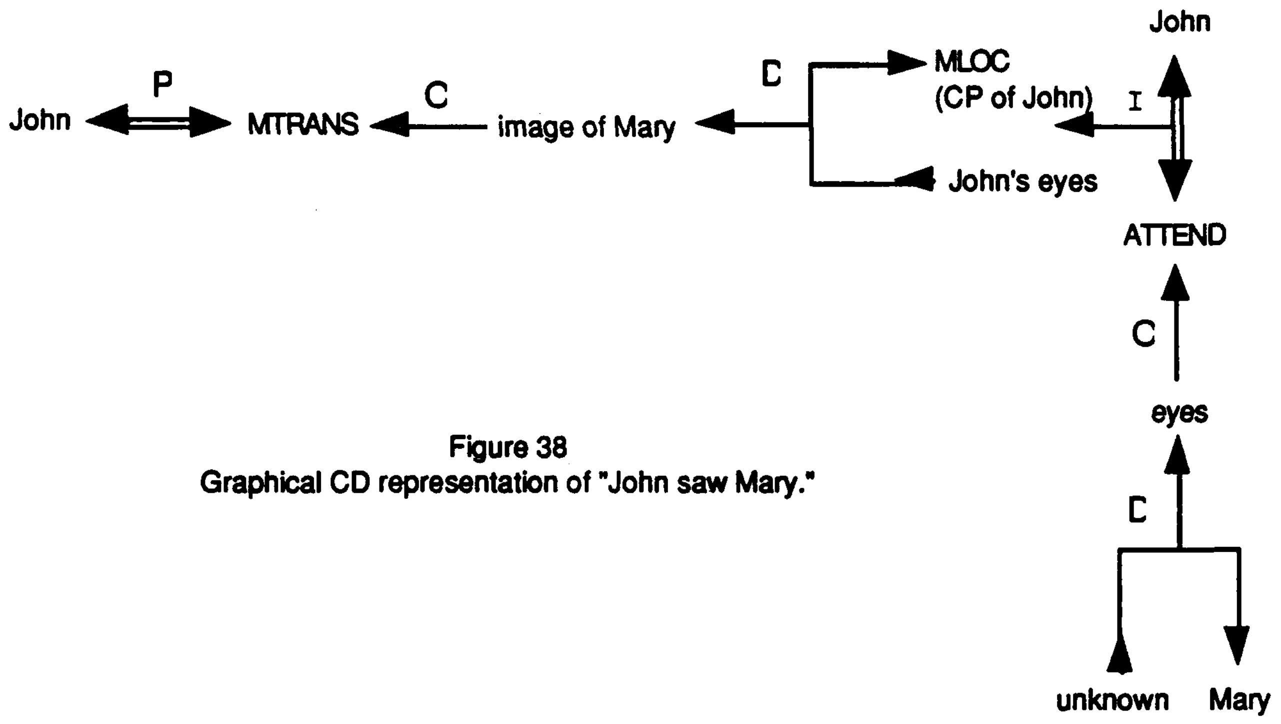


Figure 38
Graphical CD representation of "John saw Mary."

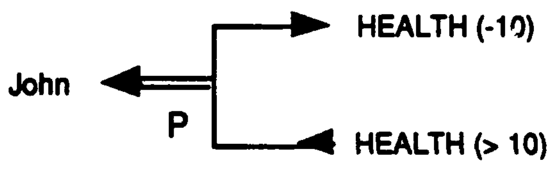


Figure 39
Graphical CD representation of "John died."

Note from the representation in Figure 39 of "John died" that verbs are not necessarily represented by primitive ACTs.

CD primitive ACTs and STATEs form represention of actions. Other knowledge structures, based on CD primitives and demonstrated in computer implementations, have been developed to represent other types of information.

SAM (Script Applier Mechanism) [Cullingford; 1981], for example, uses scripts to represent stereotypical human activities such as dining out or grocery shopping. Scripts describe the sequence of events that can be expected to occur during such activities. "Dining out", for instance, can be described by the sequential occurence of the events listed in (44):

(44) 1. The customer goes to the restaurant.
2. The customer goes to a table.
3. The server brings a menu.
4. The customer orders a meal.
5. The server brings the meal.
6. The customer eats the meal.
7. The server brings the check.
8. The customer leaves a tip for the server.
9. The customer gives money to the cashier for the meal.
10. The customer leaves the restaurant.

When told "John went to a restaurant. He ordered chicken. He left a big tip.", SAM assumes that the intervening activities of the restaurant script occured.

Of course, not all human experiences can be described by scripts. PAM (Plan Applier Mechanism) [Wilensky; 1981] investigated the use of "plans", "goals", and "themes" for story understanding. Plans describe steps toward achieving a goal. If a character's goal is to satisfy his/her hunger, an appropriate plan might be to apply the restaurant script. Themes provide general information upon which predictions can be made about an individual's goals. Hunger, Love, and Success are examples of themes. Each may be ascribed to an individual, which would then help to explain that individual's goals.

SAM, PAM, and a number of other programs designed to experiment with the use of CD knowledge structures to handle various aspects of human understanding are described in [Schank and Riesbeck; 1981].

BORIS, described in [Dyer; 1983], integrates script, plans, goals, themes, and other knowledge structures into a program which reads and answers questions about short stories in a limited domain. Each of these references contains miniature versions of the programs described.

D. Word Expert Semantics

At the time of the original version of this report, there were pockets of strong interest in, and advocacy as a linguistic theory (a theory of how humans understand language) of, Word Expert Parsing (WEP) Theory. In WEP Theory, the individual contributions of the words of a sentence fragment determine the overall meaning of the fragment. In the years since the original version of this report was published however, the interest in WEP has waned.

Although "pure" WEP Theory is no longer widely considered a viable approach to Natural Language Understanding, it is included here for its historical significance and due to the fact that its contributions to NLP research and theory have not been forgotten altogether. This is evidenced in part by the recent publication of Word Expert Semantics: An Interlingual Knowledge-Based Approach [Papegaaij; 1986], in which WEP Theory is discussed and description is provided of a Machine Translation system having aspects of WEP Theory integrated into it.

Word Expert Parsing (WEP) Theory

According to WEP Theory, each word has an associated word expert which determines the meaning of the word in the context of other words. A word expert `knows' everything there is to know about a word: different senses of the word, and how the meaning of the word effects and is effected by the other words in a sentence. As each word is examined sequentially during the parsing process, its "word expert" is retrieved from memory and executed. Word experts, which can be thought of as the "code" for determining the contribution of words in context, ask questions and exchange information with other word experts, and with higher-order system processes, until all of the word experts for a particular fragment of text come to a collective agreement on the meaning of the fragment.

An interesting sidelight to WEP Theory is that it does not accept the notion of an idiom. In other parsing theories, the words of an idiom must be viewed as a single unit in order to determine the idiom's meaning. So, the meanings of "throw in the towel" and "kicked the bucket", each as a unit, are specially stored and accessed. In WEP Theory, each word expert "knows" what its word's contribution will be even within the context of the words that make up the idiom. Therefore, the word expert for "throw" contains the information necessary to understand "throw in the towel." The disambiguation of idiomatic expressions will not differ significantly from the comprehension of any other sentence fragment.

In the prototype LISP (programmed) implementation of WEP theory, each word expert uses a word sense discrimination network (or sense net) to disambiguate the role of a word in a sentence. Each node of the discrimination net represents a multiple-choice question that the word expert can ask in order to determine the single appropriate sense of the word. An example network for the word "deep" is shown in Figure 40.

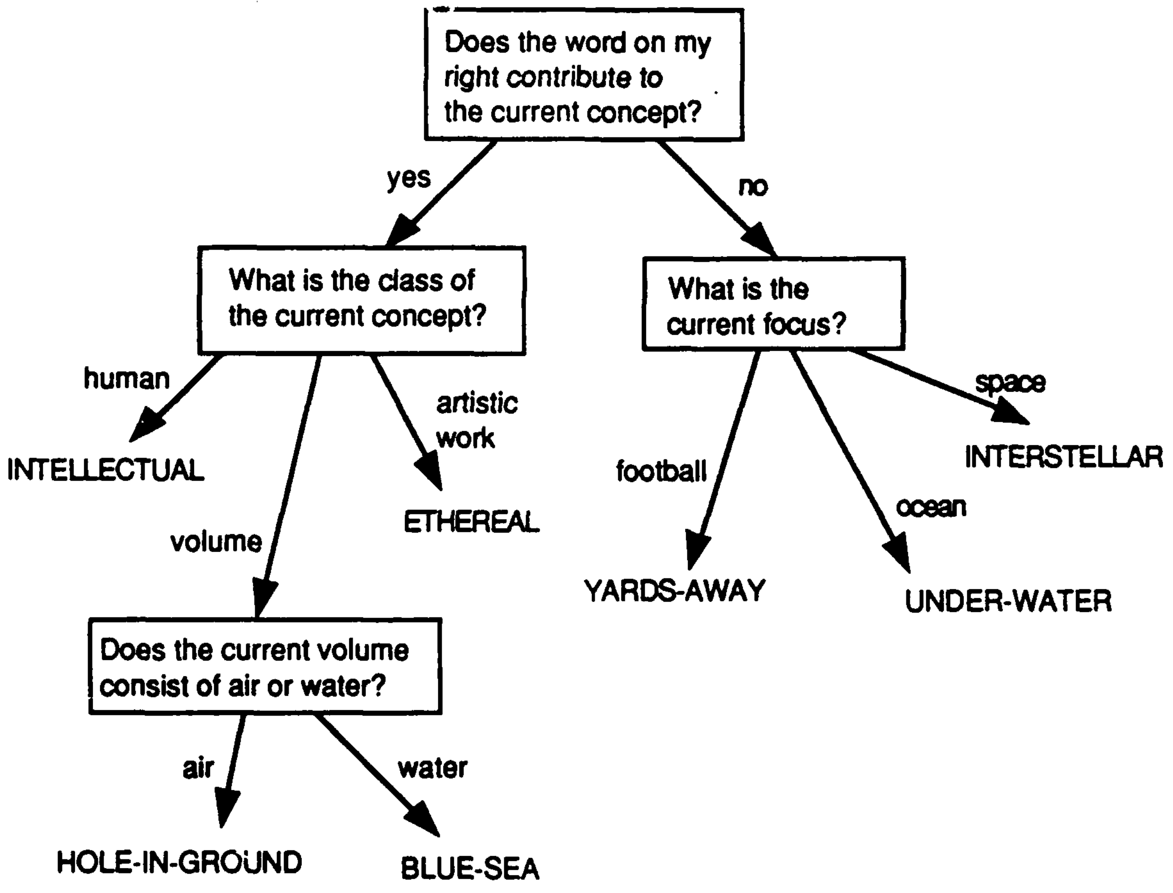

Does the word on my right contribute to the current concept?
yes
no
What is the class of the current concept?
What is the current focus?
human
INTELLECTUAL
artistic work
ETHEREAL
volume
Does the current volume consist of air or water?
air
HOLE-IN-GROUND
water
BLUE-SEA
football
YARDS-AWAY
ocean
UNDER-WATER
space
INTERSTELLAR

Figure 40
Word Sense Discrimination Network for "deep"

VI. Natural Language Generation (NLG)

For many years, the language generation task was not considered an important research area. It was thought to be much less difficult than language understanding, and less important because there was not much for computer programs to say. Now however, expert system applications are being made in language intensive fields such as medicine, military command and control, and tutoring. Quality Natural Language Generation has become necessary in order to provide an adequate ability to explain conclusions and reasoning steps.

Present-day language generation research is divided into two areas. In the Planning phase, the task is to "decide WHAT to say." Planning is often further broken down into Content Determination ("what to say") and Text Planning ("the order in which to say it"). The research described in [McKeown; 1985] is focused on the Planning phase of NLG. McKeown developed the generation system TEXT to study human communication strategies. In particular, the system is used to study discourse strategies and mechanisms for focus. Discourse strategies are general techniques people use to decide what to say next in order to get their point across in a conversation. Focus mechanisms attempt to track concepts that are central (ie. currently "in focus") in a conversation in order to constrain the number of things that may be talked about next.

The second phase of NLG is the Realization phase. In this phase, a representation of the information to relay is assumed to be present and the task is to "decide HOW to say it." Word selection, including choosing word forms, is done in the Realization phase.

Realization techniques can be divided into three classes: Canned Text, Direct Replacement, and Grammar-Based Techniques.

Canned text, the simplest approach for text realization, is exemplified by ELIZA (described in Section V.A.). Canned text is useful in systems in which the statements that the software needs to make are known, in full detail, at the time the interface is defined. The number of such statements is usually small or made up of groups of statements that are similar except for a few words. Canned textual outputs are stored strings of words or stored strings with simple word substitutions. Software compilers use Canned text to produce error messages.

In Direct Replacement, a knowledge structure is selected, or created, as the appropriate information to relay, and translated into human language using a Direct Replacement Algorithm similar to the following:

DIRECT REPLACEMENT ALGORITHM:

```
For each element of the list:

    If it is a string
    then print it,

    If it is a symbol
    then apply the symbol's generation rule and replace the symbol with the result,

    If it is a function call
    then execute the function and replace it, in the output, with the result.

Repeat until only a string of words remains.
```

The generation rules of Figure 41 and the function TENSE-FUNCTION of Figure 42 can be used to illustrate Direct Peplacement generation applied to "say" the meaning of the semantic network of Figure 43. When applying a generation rule, any element in angle brackets is replaced by the symbol (or string) found by following the arc having that element's name. Also, curly brackets represent elements that are generated only if all of the roles in the element are present.

Ruleset:

```
Rule 1.  G(BUY) =          <AGENT>,
                           TENSE-FUNCTION(BUY, <TENSE>),
                           <THEME>,
                           {"from" <POSS>},
                           {"for" <CO-THEME>}

Rule 2.  G(PERSON) =       <NAME> or "someone"

Rule 3.  G(OBJECT) =       <TYPE> or "something"

Rule 4.  G(BOOK) =         "a" {<COLOR>} "book" {"by" <AUTHOR>}
```

Figure 41

Function TENSE-FUNCTION:

```
TENSE-FUNCTION(X, Y) =
        if (X = BUY)
        then if (Y = present) then "buy"
             else if (Y = past) then "bought"
```

Figure 42

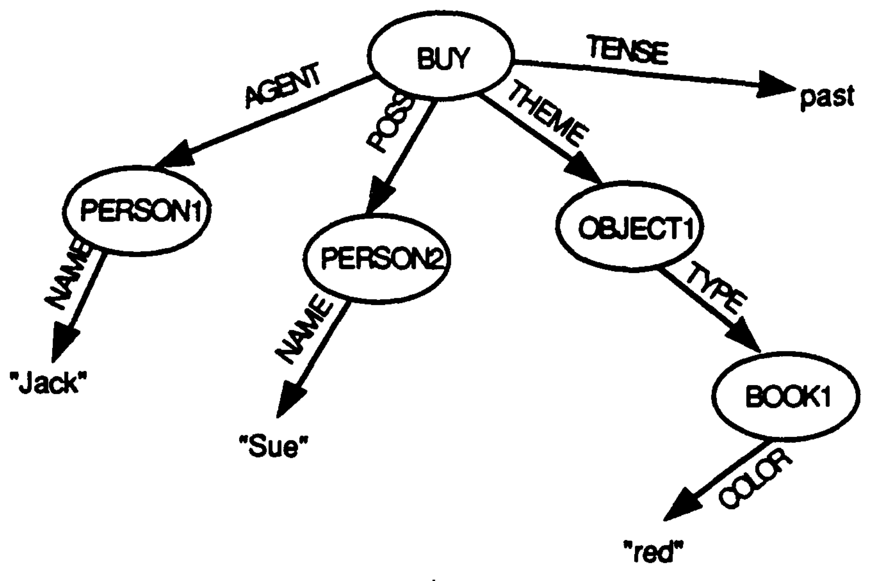


Figure 43

To generate the content of the symbol BUY in Figure 43, first apply Rule 1:

Step 1. Apply Rule 1:

```
G(BUY) =   PERSON1,
           TENSE-FUNCTION(BUY, past),
           OBJECT1,
           "from" PERSON2
```

When Rule 1 is applied, <AGENT>, <TENSE>, <THEME>, and <POSS> become PERSON1, past, OBJECT1, and PERSON2, respectively. Note that the optional phrase containing <CO-THEME> is removed because there is no CO-THEME arc in the network.

The Direct Replacement Algorithm now operates on each element of the result (PERSON1, TENSE-FUNCTION(BUY, past), OBJECT1, "from", and PERSON2) one at a time.

When Rule 2 is applied to PERSON1, TENSE-FUNCTION is evaluated with values BUY and 'past', Rule 3 is applied to OBJECT1, and Rule 2 is applied to PERSON2. The result is:

"Jack bought" BOOK1 "from Sue"

Applying Rule 4 to BOOK1 results in:

"Jack bought a red book from Sue"

Grammar-based techniques for Realization direct the generation process with grammar rules. Augmented Transition Networks are historically popular as grammar generators. One of the earliest grammar-based language generation systems ([Simmons and Slocum; 1972]) used an ATN to generate sentences from semantic networks. [Goldman; 1974] describes the use of an ATN to generate language from a Conceptual Dependency knowledge representation.

David McDonald's work in developing a program called MUMBLE ([McDonald; 1984]) is a grammar-based, domain-independent, portable tool for generating English sentences from varying representations of knowledge. Interfacing MUMBLE to an Expert System requires the definition of functions that translate a knowledge structure into a "realization specification" for MUMBLE. The trick in defining the functions is to make them as general as possible so that a relatively small number of functions are able to work together to create specifications from a large number of knowledge structures. That task is considered to be part of the Planning phase of NLG. MUMBLE "realizes" the specification shown on the next page as "Fluffy is chasing a little mouse."

```
(discourse-unit
  :head  (general-clause
          :head     (chase
                      (general-np
                        :head      (np-proper-name  "Fluffy")
                        :accessories (:number singular
                                      :gender male
                                      :determiner-policy  no-determiner))
                      (general-np
                        :head      (np-common-noun "mouse")
                                   :accessories (:number singular
                                                 :gender neuter
                                                 :determiner-policy  indefinite)
                        :further-specifications
                          (:attachment-function  restrictive-modifier
                                   :specification (predication_to-be
                                                   (adjective "little"))))
          :accessories (:tense-modal present
                        :progressive
                        :unmarked)))
```

Figure 44
MUMBLE Input Specification for
"Fluffy is chasing a little mouse."

McKeown's TEXT system and McDonald's MUMBLE have been integrated, with a parsing component called RUS (a predecessor of IRUS-86), to form RTM (RUS-TEXT-MUMBLE) ([UPENN; 1986]). RTM accepts a limited number of requests in English for definitions of, descriptions of, or comparisons between terms in a military database, formulates appropriate responses, and outputs those responses in English.

VII. Commercially-Available Natural Language Processing Systems

Natural Language Processing technology has achieved a degree of domain-independence that makes portable interfaces feasible for some applications. Portability to various back-end domains allows such systems to be useful as commercial products. Language translation, database access, and (to a lesser degree) expert systems, are applications for which commercial interface systems are available. Of the systems described in this section, INTELLECT, PARLANCE, and Language Craft have trademark protection.

Machine Translation

SYSTRAN was one of the first Machine Translation systems to be marketed. There are currently about 20 translators at the Commission of the European Communities (CEC) in Luxembourg who use SYSTRAN for routine translation, followed by human post-editing, of about 1000 pages of text per month in English-to-French, French-to-English, and English-to-Italian translations. Large dictionaries, having over 100,000 entries, appear to play a key role in the success of those applications. General Motors of Canada uses SYSTRAN for English-to-French and English-to-Spanish translations of various manuals, such as vehicle service manuals. The productivity of human translators has been reported to have been sped-up by a factor of three or four with SYSTRAN in use.

TAUM-METEO is considered the world's only example of a fully-automatic MT system. The system scans the nationwide weather communication network of the Canadian Meteorological Center (CMC) for English weather reports, translates them into French, and sends the translations back out over the communications network automatically. The system detects its own errors and passes offending input to human editors. Output deemed correct by METEO is dispatched without human involvement. With an input of over 24,000 words per day, the system correctly translates 90-95% and shuttles the other 5-10% to human translators. Although TAUM-METEO operates in a constrained domain and is not extensible or portable, similar systems could be built for similarly restricted domains.

The current state of MT and MAT systems, while not producing output as good as human translation, saves time and money. Future improvements in quality and cost reductions can be expected to improve the state of automatic language translation systems.

NLMenu (or NaturalLink) is marketed for personal computer use by Texas Instruments, Inc. as a Natural Language front-end for user databases. The NLMenu screen displays one or more menus to a user, each menu offering a string of words or a class of domain objects. An interface user selects an item from one of the menus to form the next segment of the input statement, then a new set of menus, having appropriate 'next' selections, appears. A user constructs a query by selecting items from consecutive menus. At any time, the available choices are all appropriate. As a result, any query constructed with the system is guaranteed to be within the limits of the system's grammar. User errors of typing, spelling, and improperly phrased question are eliminated, and users do not have to guess at the extent of the domain coverage. NLMenu represents an interesting, practical, and unique media for querying databases. It may serve as a viable means of communicating with software systems in many appropriate situations, but its characterization as a Natural Language interface is questionable. It is mentioned here because it is marketed as Natural Language software.

The first major commercial NL interface for databases, INTELLECT (formerly called ROBOT [Harris; 1977]), is produced by the Artificial Intelligence Corporation in Waltham, Massachusetts [AI Corporation; 1980]. INTELLECT provides access to a number of Database Management Systems on IBM, Honeywell, and Prime computers. Initial versions of the system had to be tailored to the user's domain with several weeks of effort by the AI Corporation (AIC) for each installation. More recently tailoring is done by systems staff in user installations with the aid of documentation and training provided by AIC.

The Symantec Corporation of Cupertino, California, produces Q&A ([Hendrix; 1986]) for Natural Language database access on the IBM personal computer and its compatibles. Q&A is a descendent of NL research conducted at SRI International in the 1970's that produced the LADDER system ([Hendrix, et al; 1978]).

The most recent arrival on the database interface commercial scene is PARLANCE from BBN, Inc. ([BBN, Inc.; 1988]). As part of the marketing fee for the system, BBN configures PARLANCE to a user's relational database. The application of PARLANCE to a Navy database is described in [Bates; 1989].

Language Craft ([Carnegie Group, Inc.; 1985]), marketed by Carnegie Group, Inc., Pittsburgh, Pennsylvania, is used to construct Natural Language Interfaces to databases and expert systems. Language Craft evolved from work at Carnegie Mellon University that developed the XCALIBUR system. XCALIBUR provides NL access to an expert system that configures VAX^{TM} systems to customer specifications. The Carnegie Group provides specialized training for people wishing to use Language Craft to develop interfaces. To develop an interface, "caseframes" are designed which are matched against user inputs. The caseframe example below outlines possible ways of requesting a map display system to zoom-in on an area of the map. "*panes*" is an identifier for a caseframe which, in this instance, describes display panes on a screen-displayed map.

```
[*zoom-in*
     :type    sentential
     :header   zoom in !! go in
     :cases
          (levels
               :case-marker   by
               :filler     $n (level !! levels))
          (map
               :case-marker   on
               :filler      *panes*)]
```

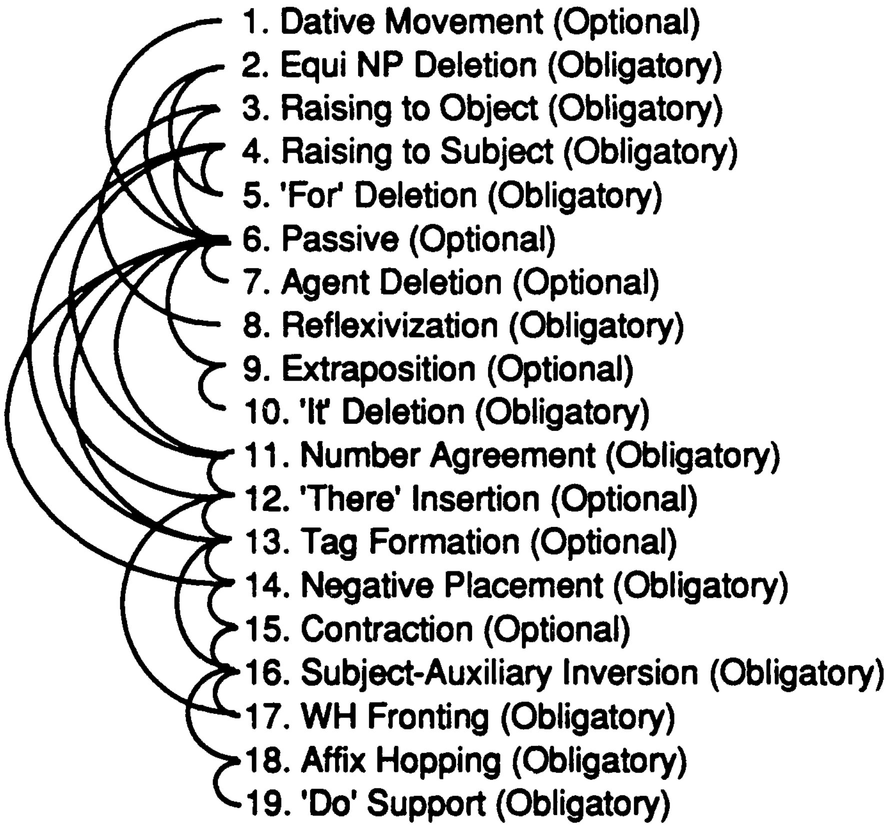
The header contains key words that identify a match with this caseframe template. Cases are optional segments that may be included in an input. For each 'case' the case-marker is a keyword to match and the filler is a pattern of words that is to follow that keyword. Each caseframe is defined by the interface builder to allow correct matching of a large number of possible user inputs. The caseframe shown can match all of the inputs shown below and more. Additional rules and patterns broaden the breadth of understood user inputs and provide response generation capabilities that are also based on matched patterns.

Zoom in by 3 levels on the display pane.
By 1 level, go in on the map pane.
Zoom in on the display pane.
By 2 levels on the display pane, zoom in.
Zoom in.

Cognitive Systems, Inc. of New Haven, Connecticutt custom-builds NL database interfaces, based on Conceptual Dependency Theory. An example of their work is the EXPLORER system developed for a major oil company. EXPLORER retrieves information from a database containing approximately 175 fields of information on over one million oil wells, as well as information on most of the geological formations in North America, and produces maps displaying geological data and oil well information ([Cognitive Systems, Inc.; undated], [Shwartz; 1984]).

APPENDIX A
Transformational Rule Ordering

The following is the ordered list of transformations of the Standard Theory of Transformational Grammar, from [Akmajian and Heny; 1975]. A line connecting two rules indicates that an explicit argument for their displayed ordering exists. Although there is not an argument for every pair, the combination of all of the pairwise orderings leads to a total ordering.



1. Dative Movement (Optional)
2. Equi NP Deletion (Obligatory)
3. Raising to Object (Obligatory)
4. Raising to Subject (Obligatory)
5. 'For' Deletion (Obligatory)
6. Passive (Optional)
7. Agent Deletion (Optional)
8. Reflexivization (Obligatory)
9. Extraposition (Optional)
10. 'It' Deletion (Obligatory)
11. Number Agreement (Obligatory)
12. 'There' Insertion (Optional)
13. Tag Formation (Optional)
14. Negative Placement (Obligatory)
15. Contraction (Optional)
16. Subject-Auxiliary Inversion (Obligatory)
17. WH Fronting (Obligatory)
18. Affix Hopping (Obligatory)
19. 'Do' Support (Obligatory)

APPENDIX B
Sample Dialogue with SOPHIE
(A SOPHisticated Instructional Environment)

SOPHIE [Brown; 1974] is a computer program designed to tutor students on the principles of electronic circuit debugging. The following is a sample of the dialogue that may occur between a student and the system:

SOPHIE: THE INSTRUMENT HAS A FAULT AND ITS PRESENT CONTROL SETTINGS ARE:

CC	1.0	(CURRENT CONTROL - FROM 0.0 (MIN CURRENT) TO 1.0)
CR	HIGH	(CURRENT RANGE SWITCH, HIGH = 1 AMP, LOW = .1 AMP)
LOAD	20	(LOAD RESISTANCE IN OHMS)
VC	1.0	(VOLTAGE CONTROL - FROM 0.0 9MIN VOLTAGE) TO 1.0)
VR	HIGH	(VOLTAGE RANGE SWITCH, HIGH = 30 VOLTS, LOW = 10 VOLTS)

Student: What is the output?
SOPHIE: THE OUTPUT VOLTAGE IS 11.7 VOLTS.

Student: What is it in a working instrument?
SOPHIE: IN A WORKING CIRCUIT THE OUTPUT VOLTAGE IS 19.9 VOLTS.
THAT VALUE IS INCORRECT.
(Note the student's use of pronomial reference. "IT" is used to refer to the output voltage.)

Student: What is the voltage between R8 and Q1?
SOPHIE: THE VOLTAGE AT N1 IS 47.6 VOLTS.

Student: What is the BSE emitter voltage of the voltage limiting transistor?
SOPHIE: =BASE
=TRANSISTOR
THE BASE EMITTER VOLTAGE OF Q5 IS -.037 MILLIVOLTS.
(The spelling corrector has detected and corrected two misspellings.)

Student: What about for the current limiting transistor?
SOPHIE: THE BASE EMITTER VOLTAGE OF Q6 IS .349 VOLTS.
(The student is actually asking, "What about the base emitter voltage for the current limiting transistor?" This is an example of the use of ellipsis.)

APPENDIX C
Definite Clause Grammar

A Context-Free Gramma:::

$$\text{sentence} \longrightarrow \text{noun\_phrase, verb\_phrase.}$$

$$\text{noun\_phrase} \longrightarrow \text{determiner, noun.}$$

$$\text{verb\_phrase} \longrightarrow \text{verb.}$$
$$\text{verb\_phrase} \longrightarrow \text{verb, noun\_phrase.}$$

$$\text{determiner} \longrightarrow \text{[the].}$$

$$\text{noun} \longrightarrow \text{[man].}$$
$$\text{noun} \longrightarrow \text{[apple].}$$

$$\text{verb} \longrightarrow \text{[eats].}$$
$$\text{verb} \longrightarrow \text{[sings].}$$

Given the context-free grammar above, Prolog's Grammar Rule Interpreter would rewrite it into the Prolog code shown below. This example is from [Clocksin and Mellish; 1981].

Prolog code for a small grammar:

```
sentence(S0,S) :- noun_phrase(S0,S1), verb_phrase(S1,S).

noun_phrase(S0,S) :- determiner(S0,S1), noun(S1,S).

verb_phrase(S0,S) :- verb(S0,S).
verb_phrase(S0,S) :- verb(S0,S1), noun_phrase(S1,S).

determiner([the|S],S).

noun([man|S],S).
noun([apple|S],S).

verb([eats|S],S).
verb([sings|S],S).
```

APPENDIX D
Conceptual Dependency (CD) Primitive ACTs

Listed and briefly described here, are the primitive ACTs defined for Conceptual Dependency Theory. Each of them is described in-depth in [Schank; 1975] and [Dyer; 1983].

ATRANS: Represents the transfer of an abstract relationship, such as possession. One sense of the word "give" is to ATRANS something to someone else. To "take" is to ATRANS something to oneself. ATRANS is also used in conceptualizations of "bought" and "sold".

PTRANS: Represents the transfer of the physical location of an object. For example, "go" is a PTRANS of oneself from one place to another. "Driving" and "flying" are also represented by PTRANS.

MTRANS: Represents a transfer of mental information from one place to another, for instance from one individual to another as in "speaking" and "talking", or among distinct parts of one individual's memory as in "remembering", "forgetting", and "learning".

MBUILD: An action which creates new conceptual structures from old ones. Examples are "concluding" and "realizing".

PROPEL: The application of a physical force, as in "hitting", "falling", and "pulling".

MOVE: The movement of a bodypart of an animate organism. The conceptualizations of "waving" and "dancing" use the MOVE primitive.

INGEST: When an organism takes something from outside itself and makes it internal, it INGESTs it. "Breathing" and "smoking" are acts of INGESTing.

EXPEL: The opposite of INGEST, an organism EXPELs when it takes something from inside itself and makes it external, as in "sweating" and "crying".

GRASP: To physically contact an object. Examples are "grabbing" and "hugging".

SPEAK: Any vocalization. "Squeak" and "quack" would be included as well as "say".

ATTEND: The act of directing a sense organ. To "hear" something involves ATTENDing one's ears toward the sounds being made (note, not towards the being or thing making the sounds).

APPENDIX E
Terminology Glossary

APPENDIX E (Continued)
Terminology Glossary

Terminology Glossary

STATEs...58-62
surface cases...51
surface structure...26-27, 29-31, 32, 34
syntactic cases...51
syntax...1, 3, 7, 11-13, 18-48
Text Understanding...5-6
token nodes...56
top-down processing...13-14
Trace Theory...34-35
traces...34-35
transformations...29-30, 31, 32, 34, 35, 40, 41
Transformational Generative Grammar...26
type (0, 1, 2, 3) grammars...19-24
type nodes...56
underlying phrase markers...29
unrestricted grammars...20
word experts...63
Word Expert Parsing (WEP) Theory...63-64
word sense discrimination network...64

APPENDIX F
Software Systems Glossary

REFERENCES

[Akmajian, et al; 1984]
Akmajian, Adrian, Richard A. Demers and Robert M. Harnish, Linguistics: An Introduction to Language and Communication, Cambridge, MA: The MIT Press, 1984.

[Akmajian and Heny; 1975]
Akmajian, Adrian and Frank W. Heny, An Introduction to the Principles of Transformational Syntax, Cambridge, MA: The MIT Press, 1975.

[Allen; 1987]
Allen, James, Natural Language Understanding, Menlo Park, CA: The Benjamin/Cummings Publishing Company, Inc., 1987.

[AI Corporation; 1980] Artificial Intelligence Corporation, INTELLECT User's Guide, Waltham, MA, 1980.

[Appelt; 1985]
Appelt, Douglas E., Planning English Sentences, New York: Cambridge University Press, 1985.

[Bar-Hillel; 1960]
Bar-Hillel, Y, "The Present Status of Automatic Translation of Languages", in F.L. Alt (Ed.), Advances in Computers, Vol. 1, New York: Academic Press, 1960.

[Bar-Hillel; 1971]
Bar-Hillel, Y, "Some Reflections on the Present Outlook for High-Quality Machine Translation", In Lehman, W.P. and R. Stachowitz (Eds.), Feasibility Study on Fully Automatic High Quality Translation, Linguistics Research Center, University of Texas at Austin, Final Technical Report RADC TR-71-295, 1971.

[Barr and Feigenbaum; 1981]
Barr, Avron and Edward Feigenbaum (Eds.), The Handbook of Artificial Intelligence: Volume 1, Los Altos, CA: William Kaufmann, Inc., 1981.

[Barton, et al;1987]
Barton, G. Edward, Robert C. Berwick, and Eric Sven Ristad, Computational Complexity and Natural Language, Cambridge, MA: The MIT Press, 1987.

[Bates and Bobrow; 1984]
Bates, Madelaine and Robert J. Bobrow, "Natural Language Interfaces: What's Here, What's Coming, and Who Needs It", In Reitman, Walter (ed.), Artificial Intelligence Applications for Business, Norwood, NJ: Ablex Publishing, 1984.

[Bates; 1989]
Bates, Madelaine, "Rapid Porting of the PARLANCE Natural Language Interface," Proceedings of the DARPA Speech and Natural Language Workshop, Philadelphia, PA, February 1989.

[BBN, Inc.; 1988]
BBN, Inc., "The PARLANCE Database Interface from BBN Laboratories, Inc.," Marketing Leaflet, 1988.

[Berwick and Weinberg; 1986]
Berwick, Robert C. and Amy S. Weinberg, The Grammatical Basis of Linguistic Performance, Cambridge, MA: The MIT Press, 1986.

[Brown, et al; 1974]
Brown, J.S., R.R. Burton, and A.G. Bell, "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting", BBN Report Number 2790, Cambridge, MA: Bolt, Beranek and Newman, Inc., 1974.

[Bruce; 1975]
Bruce, Bertram C., "Case Systems for Natural Language", Artificial Intelligence, Volume 6, Number 4, 1975, pges. 327-360.

[Bruce; 1983]
Bruce, Bertram C., "Belief Systems and Language Understanding", In Sedelow, Walter A. and Sally Yeates Sedelow (Eds.), Computers and Language Research 2, Berlin: Walter de Gruyter & Co., 1983.

[Brady and Berwick; 1984]
Brady, Michael and Robert C. Berwick, Computational Models of Discourse, Cambridge, MA: The MIT Press, 1984.

[Burns and Capps; 1988]
Burns, Hugh L. and Charles G. Capps, "Foundations of Intelligent Tutoring Systems: An Introduction", In Polson and Richardson (Eds.), Foundations of Intelligent Tutoring Systems, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[Carnegie Group, Inc.; 1985]
Carnegie Group, Inc., "Language Craft," Language Craft Tranining Materials, 1985.

[Charniak; 1976a]
Charniak, Eugene, "Inference and Knowledge: Part 1", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Charniak; 1981]
Charniak, Eugene, "Six Topics in Search of a Parser: An Overview of AI Language Research", Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Volume 2, 1981, pg. 1079-1087.

[Charniak; 1976b]
Charniak, Eugene, "Syntax in Linguistics", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Charniak, et al; 1980]
Charniak, Eugene, Christopher K. Riesbeck and Drew V. McDermott, Artificial Intelligence Programming, Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.

[Charniak and Wilks; 1976]
Charniak, Eugene and Yorick Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Chomsky; 1957]
Chomsky, Noam, Syntactic Structures, The Hague: Mouton, 1957.

[Chomsky; 1965]
Chomsky, Noam, Aspects of the Theory of Syntax, Cambridge, MA: The MIT Press, 1965.

[Chomsky; 1977]
Chomsky, Noam, "On Wh-movement", In P. Culicover, T. Wasow, and A. Akmajian (eds.), Formal Syntax, New York: Academic Press, 1977.

[Cleaveland and Uzgalis; 1977]
Cleaveland, J. Craig and Robert C. Uzgalis, Grammars for Programming Languages, New York: North-Holland, 1977.

[Clocksin and Mellish; 1981]
Clocksin, W.F. and C.S. Mellish, Programming in Prolog, Berlin: Springer, 1981.

[Cullingford; 1981]
Cullingford, Richard, "SAM", In Schank and Riesbeck (Eds.), Inside Computer Understanding: Five Programs Plus Miniatures, Hillsdale, NJ: Lawrence Erlbaum Associates, 1981, pges. 75-119.

[Dowty; 1982]
Dowty, David, "Grammatical Relations and Montague Grammar", In Jacobson, Pauline I. and Geoffrey K. Pullum (Eds.), The Nature of Syntactic Representation, Holland: Reidel Publishing, 1982.

[Dyer; 1983]
Dyer, Michael George, In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension, Cambridge, MA: The MIT Press, 1983.

[Earley; 1970]
Earley, Jay, "An Efficient Context-Free Parsing Algorithm", Communications of the ACM, Vol. 13, Number 2, February 1970.

[Fillmore; 1968]
Fillmore, Charles, "The Case for Case", In Bach and Harms (Eds.), Universals in Linguistics Theory, New York: Holt, Rinehart and Winston, 1968, pgs. 1-90.

[Frankline, et al; 1987]
Frankline, J., Laura Davis, Randall Shumaker, and Paul Morawski, "Automated Natural-Language Understanding of Military Messages", In Shapiro, Stuart C. (Ed.), Encyclopedia of Artificial Intelligence, New York, John Wiley & Sons, 1987.

[Gazdar; 1982]
Gazdar, Gerald, "Phrase Structure Grammar", In Jacobson, Pauline and Geoffrey K. Pullum (Eds.), The Nature of Syntactic Recognition, Holland: Reidel Publishing, 1982.

[Gazdar; 1983a]
Gazdar, Gerald, "NLs, CFLs and CF-PSGs", In Karen Sparck Jones and Yorick Wilks (Eds.), Automatic Natural Language Parsing, England: Ellis Horwood Limited, 1983, pg. 81-93.

[Gazdar; 1983b]
Gazdar, Gerald, "Phrase Structure Grammars and Natural Languages", Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Volume 1, 1983, pg. 556-565.

[Gevarter; 1983]
Gevarter, William B., "An Overview of Computer-Based Natural Language Processing", NASA Technical Memorandum 85635, 1983.

[Grishman and Hirschman; 1986]
Grishman, Ralph and Lynette Hirschman, "PROTEUS and PUNDIT: Research in Text Understanding", In The FINITE STRING Newsletter: Site Reports from Several Natural Language Technology Base Contracts with DARPA's Strategic Computing Program, Computational Linguistics, Vol. 12, Number 2, April-June 1986.

[Grosz, et al; 1986]
Grosz, Barbara J., Karen Sparck Jones, and Bonnie Lynn Webber (ed.), Readings in Natural Language Processing, Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.

[Grosz and Sidner; 1986]
Grosz, Barbara and Candace Sidner, "Attention, Intention, and the Structure of Discourse", Computational Linguistics, Vol. 12, Number 3, 1986.

[Harris; 1977]
Harris, Larry R., "User Oriented Database Query with the ROBOT Natural Language Query System", Proceedings of the Third International Conference on Very Large Data Bases, 1977.

[Hayes; 1976]
Hayes, Philip, "Semantic Markers and Selectional Restrictions", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Hendrix, et al; 1978]
Hendrix, G.G., E.D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data," ACM Transactions on Database Systems, Vol. 3, 1978.

[Hendrix; 1986]
Hendrix, Gary G., "Bringing Natural Language Processing to the Microcomputer Market: The Story of Q&&A", Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics, 10-13 June 1986.

[Hobbs; 1978]
Hobbs, Jerry R., "Making Computational Sense of Montague's Intensional Logic", Artificial Intelligence, 9, 1978, pg. 287 - 306.

[Hopcroft and Ullman; 1979]
Hopcroft, John E. and Jeffrey D. Ullman, Introduction to Automata Theory, Languages, and Computation, Reading, MA: Addison-Wesley Publishing Company, 1979.

[Jones and Wilks; 1983]
Jones, Karen Sparck and Yorick Wilks, Automatic Natural Language Parsing, England: Ellis Horwood Limited, 1983.

[Jones and Warren; 1982]
Jones, Mark A. and David S. Warren, "Conceptual Dependency and Montague Grammar: A Step Toward Conciliation", Proceedings of the American Association or Artificial Intelligence, 1982, pg. 79 - 82.

[Katz; 1966]
Katz, Jerrold J., The Philosophy of Language, New York: Harpers & Row, 1966.

[Katz and Postal; 1964]
Katz, Jerrold J. and Paul M. Postal, An Integrated Theory of Linguistic Descriptions, Cambridge, MA: The MIT Press, 1964.

[Kay; 1973]
Kay, Martin, "The MIND System", In Rustin, R. (Ed.), Natural Language Processing, Englewood Cliffs, NJ: Prentice-Hall, 1973.

[Kempson; 1977]
Kempson, Ruth M., Semantic Theory, New York: Cambridge University Press, 1977.

[King; 1976]
King, Margaret, "Generative Semantics", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[King; 1983]
King, Margaret (Ed.), Parsing Natural Language, New York: Academic Press, 1983.

[Lehnert; 1978]
Lehnert, Wendy G., The Process of Question-Answering: A Computer Simulation of Cognition, Hillsdale, NJ: Lawrence Erlbaum Associates, 1978.

[Lehrberger and Bourbeau; 1988]
Lehrberger, John and Laurent Bourbeau, Machine Translation: Linguistic Characteristics of MT Systems and General Methodology of Evaluation, Philadelphia, PA: John Benjamins Publishing Company, 1988.

[Levelt; 1976]
Levelt, W.J.M., "Formal Grammars and the Natural Language User: A Review", In A. Marzollo (Ed.), Topics in Artificial Intellige ice, New York: Springer-Verlag, 1976.

[Mann, et al; 1982]
Mann, W., M. Bates, B. Grosz, and D. McDonald, "Text Generation", American Journal of Computational Linguistics, Vol. 8, Number 2, April-June 1982.

[Marcus; 1980]
Marcus, Mitch, A Theory of Syntactic Recognition for Natural Language, Cambridge, MA: The MIT Press, 1980.

[McDonald; 1984]
McDonald, David D., "Natural Language Generation as a Computational Problem An Introduction", In Brady and Berwick (Eds.), Computational Models of Discourse, Cambrdige, MA: The MIT Press, 1984.

[McDonald; 1987]
McDonald, David D., "Natural Language Generation", In Shapiro, Stuart C. (Ed.), Encyclopedia of Artificial Intelligence, New York, John Wiley & Sons, 1987.

[McDonald and Meteer; 1986]
McDonald, David D., and Marie W. Meteer, Course materials for MUMBLE tutorial to RADC participants, 1986.

[McKeown; 1985]
McKeown, Kathleen R., Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text, New York: Cambridge University Press, 1985.

[Miller; 1988]
Miller, James R., "The Role of Human-Computer Interaction in Intelligent Tutoring Systems", In Polson and Richardson (Eds.), Foundations of Intelligent Tutoring Systems, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[Milne; 1983]
Milne, Robert W., Resolving Lexical Ambiguity in a Deterministic Parser, Ph.D Thesis, University of Edinburgh, 1983.

[Palmer, et al; 1986]
Palmer, Martha S., Deborah A. Dahl, Rebecca J. Schiffman, and Lynette Hirschman, "Recovering Implicit Information", Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics, Columbia University, NY, 10-13 June 1986.

[Papegaaij; 1986]
Papegaaij, B.C., Word Expert Semantics: An Interlingual Knowledge-Based Approach, Dordrecht, The Netherlands: Foris Publications, 1986.

[Pereira; 1981]
Pereira, Fernando, "Extraposition Grammars", American Journal of Computational Linguistics, Volume 7, Number 4, 1981, pg. 243-256.

[Pereira and Shieber; 1987]
Pereira, Fernando and Stuart M. Shieber, Prolog ar.1 Natural-Language Analysis, Stanford, CA: Center for the Study of Language and Information, 1987.

[Pereira and Warren; 1980]
Pereira, Fernando and David H.D. Warren, "Definite Clause Grammars for Language Analysis: A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, 13, 1980, pg. 231-278.

[Polson and Richardson; 1988]
Polson, Martha C. and J. Jeffrey Richardson (Eds.), Foundations of Intelligent Tutoring Systems, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[Pullum; 1984]
Pullum, Geoffrey K., "On Two Recent Attempts to Show that English is Not a CFL", Computational Linguistics, Volume 10, Numbers 3-4, 1984, pges. 182-186.

[Pulman; 1983]
Pulman, S., "Trace Theory, Parsing and Constraints", In King, M. (Ed.), Parsing Natural Language, New York: Academic Press, 1983.

[Quillian; 1968]
Quillian, M. Ross, "Semantic Memory", In Minsky, M. (Ed.), Semantic Information Processing, Cambridge, MA: The MIT Press, 1968.

[Quillian; 1969]
Quillian, M. Ross, "The Teachable Language Comprehender: A Simulation Program and Theory of Language", Communications of the Association for Computing Machinery, 12, 1969, Pges. 459-476.

[Raphael; 1968]
Raphael, Bertram, "SIR: A Computer Program for Semantic Information Retieval", In Minsky, M. (Ed.), Semantic Information Processing, Cambridge, MA: The MIT Press, 1968.

[Rieger; 1979]
Rieger, Chuck and Steve Small, "Word Expert Parsing", Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 2, 1979, pg. 723-728.

[Ritchie; 1983a]
Ritchie, G. D., "Semantic Networks - A General Definition and a Survey", Information Technology: Research and Development,2, 1983, pgs. 187-231.

[Ritchie; 1983b]
Ritchie, G. D., "Semantics in Parsing", In King (Ed.), Parsing Natural Language, New York: Academic Press, 1983.

[Ritchie and Thompson; 1984]
Ritchie, Graeme and Henry Thompson, "Natural Language Processing", In O'Shea, Tim and Marc Eisenstadt (Eds.), Artificial Intelligence: Tools, Techniques, and Applications, New York: Harper & Row, 1984, pges. 358-388.

[Samlowski; 1976]
Samlowski, Wolfgang, "Case Grammar", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Sampson; 1983]
Sampson, Geoffrey R., "Context Free Parsing and the Adequacy of Context-Free Grammars", In King (Ed.), Parsing Natural Language, New York: Academic Press, 1983.

[Schank; 1975]
Schank, Roger C., Conceptual Information Processing, Amsterdam: North-Holland, 1975.

[Schank and Abelson; 1977]
Schank, Roger C. and Robert P. Abelson, Scripts, Plans, Goals, and Understanding, Hillsdale, NJ: Lawrence Erlbaum Associates, 1977.

[Schank and Lehnert; 1979]
Schank, Roger C. and Wendy Lehnert, "Review of Natural Language Processing", In Wegner (Ed.), Research Directions in Software Technology, Cambridge, MA: The MIT Press, 1979, pg. 750 - 766.

[Schank and Riesbeck; 1981]
Schank, Roger C. and Christopher K. Riesbeck (Eds.), Inside Computer Understanding: Five Programs Plus Miniatures, Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

[Scragg; 1976]
Scragg, Greg, "Semantic Nets as Memory Models", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Searle, et al; 1980]
Searle, John R., Ferenc Kiefer, and Manfred Bierwisch (Eds.), Speech Act Theory and Pragmatics, Boston, MA: D. Reidel Publishing, 1980.

[Sedlock; 1988]
Sedlock, David, "The Natural Language-Data Base Connection", AI Expert, July 1988.

[Sells; 1985]
Sells, Peter, Lectures on Contemporary Syntactic Theories, Stanford, CA: Center for the Study of Language and Information, 1985.

[Shapiro; 1987]
Shapiro, Stuart C. (Ed.), Encyclopedia of Artificial Intelligence (Volumes 1 and 2), New York: John Wiley && Sons, 1987.

[Shwartz; 1984]
Shwartz, Steven P., "Natural Language Processing in the Commercial World", In Reitman, Walter (ed.), Artificial Intelligence Applications for Business, Norwood, NJ: Ablex Publishing, 1984.

[Simmons; 1970]
Simmons, Robert F.; "Natural Language Question-Answering Systems:1969", Communications of the Association for Computing Machinery, 13, 1970, pg. 15-30.

[Simmons and Slocum; 1972]
Simmons, R. and J. Slocum, "Generating English Discourse from Semantic Networks", Communications of the Association for Computing Machinery, 15, 10, 1972.

[Sippu and Soisalon-Soininen; 1988]
Sippu, Seppo and Eljas Soisalon-Soininen, Parsing Theory 1: Languages and Parsing, New York: Springer-Verlag, 1988.

[Slocum; 1985]
Slocum, Jonathan, "A Survey of Machine Translation: Its History, Current Status, and Future Prospects", Computational Linguistics, Vol. 11, Number 1, January-March 1985, pg. 1-17.

[Small;1981]
Small, Steve, "Viewing Word Expert Parsing as Linguistic Theory", Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 1, 1981, pg. 70-76.

[Tennant; 1987]
Tennant, H., "Menu-Based Natural Language", In Shapiro, Stuart C. (Ed.), Encyclopedia of Artificial Intelligence, New York: John Wiley & Sons, 1987.

[Thompson; 1983]
Thompson, Henry, "MCHART: A Flexible, Modular Chart Parsing System", Proceedings of the National Conference on Artificial Intelligence, 1983, pg. 408-410.

[Thompson and Ritchie; 1984]
Thompson, Henry and Graeme Ritchie, "Implementing Natural Language Parsers", In O'Shea, Tim and Marc Eisenstadt (Eds.), Artificial Intelligence: Tools, Techniques, and Applications, New York: Harper && Row, 1984, pges. 245-300.

[Thorne, et al; 1968]
Thorne, J., P. Bratley, and H. Dewar, "The Syntactic Analysis of English by Machine", In Michie, D. (Eds.), Machine Intelligence, 3, New York: American Elsevier, 1968.

[UPENN; 1986]
University of Pennsylvania, Department of Computer and Information Science, "Research in Natural Language Processing", In The FINITE STRING Newsletter: Site Reports from Several Natural Language Technology Base Contracts with DARPA's Strategic Computing Program, Computational Linguistics, Vol. 12, Number 2, April-June 1986.

[Varile; 1983]
Varile, G.B., "Charts: A Data Structure for Parsing", In King (Ed.), Parsing Natural Language, New York: Academic Press, 1983.

[Walter; 1986]
Walter, Sharon M., "Natural Language Processing: A Tutorial", Rome Air Development Center (RADC), RADC Technical Report 86-110, 1986.

[Warren and Pereira: 1982]
Warren, David S. and Fernando C.N. Pereira, "An Efficient Easily Adaptable System for Interpreting Natural Language Queries", American Journal of Computational Linguistics, Volume 8, Number 3-4, 1982, pg. 110-122.

[Webber; 1979]
Webber, Bonnie Lynn, A Formal Approach to Discourse Anaphora, New York: Garland Publishing, Inc., 1979.

[Webber and Finin; 1984]
Webber, Bonnie Lynn and Tim Finin, "In Response: Next Steps in Natural Language Interaction", In Reitman, Walter (ed.), Artificial Intelligence Applications for Business, Norwood, NJ: Ablex Publishing, 1984.

[Weischedel, et al; 1989]
Weischedel, Ralph M., Robert J. Bobrow, Damaris Ayuso, and Lance Ramshaw, "Portability in the JANUS Natural Language Interface," Proceedings of the DARPA Speech and Natural Language Workshop, Philadelphia, PA, February 21-23, 1989.

[Weizenbaum; 1966]
Weizenbaum, Joseph, "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine", Communications of the Association for Computing Machinery, Volume 9, Number 1, 1966, pg. 36-45.

[Wilensky; 1981]
Wilensky, Robert, "PAM", In Schank and Riesbeck (Eds.), Inside Computer Understanding: Five Programs Plus Miniatures, Hillsdale, NJ: Lawrence Erlbaum Associates, 1981, pges. 136-179.

[Wilks; 1975]
Wilks, Yorick, "An Intelligent Analyzer and Understander of English", Communications of the Association for Computing Machinery, Volume 18, Number 5, 1975, pg. 264-274.

[Wilks; 1976a]
Wilks, Yorick, "Parsing English II", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam: North-Holland, 1976.

[Wilks; 1976b]
Wilks, Yorick, "Philosophy of Language", In Charniak and Wilks (Eds.), Computational Semantics, Amsterdam:North-Holland, 1976.

[Winograd; 1972]
Winograd, Terry, Understanding Natural Language, New York: Academic Press, 1972.

[Winograd; 1983]
Winograd, Terry, Language as a Cognitive Process, Reading, MA: Addison-Wesley, 1983.

[Woods; 1970]
Woods, W. A., "Transition Network Grammars for Natural Language Analysis", Communications of the Association for Computing Machinery, Volume 13, Number 10, 1970, pges. 591-606.

[Woods; 1978]
Woods, William A., "Semantics and Quantification in Natural Language Question Answering", In Yovits, Marshall C. (Ed.), Advances in Computers, Volume 17, New York: Academic Press, 1978, pges. 1-87.