Natural Language Question-Answering Systems: 1969

ROBERT F. SIMMONS
The University of Texas at Austin, Austin, Texas*

Recent experiments in programming natural language question-answering systems are reviewed to summarize the methods that have been developed for syntactic, semantic, and logical analysis of English strings. It is concluded that at least minimally effective techniques have been devised for answering questions from natural language subsets in small scale experimental systems and that a useful paradigm has evolved to guide research efforts in the field. Current approaches to semantic analysis and logical inference are seen to be effective beginnings but of questionable generality with respect either to subtle aspects of meaning or to applications over large subsets of English. Generalizing from current small-scale experiments to language-processing systems based on dictionaries with thousands of entries—with correspondingly large grammars and semantic systems—may entail a new order of complexity and require the invention and development of entirely different approaches to semantic analysis and question answering.

KEY WORDS AND PHRASES: question-answering system, natural language, artificial intelligence, language processing, fact retrieval, semantics
CR CATEGORIES: 3.6, 3.64, 3.7, 3.74

1. Introduction

Kuhn (1962) has persuasively argued that science progresses by means of its paradigms—its models of the general nature of a research area—and that at the frontiers of research the primary quest is for a good paradigm. The small frontier outpost of language data processing has been characterized by an intensive seeking for a paradigm suitable to guide its researchers as they survey the complex topography of natural language structures. The earliest paradigm—one that led mechanical translators and early information retrievalists into a hopeless cul-de-sac—was that words (i.e. strings of letters) are the units of meaning; that mechanical translation requires simply the discovery and substitution of target language equivalent words; that information retrieval requests and data structures can be adequately represented by some Boolean combination of words.

With each succeeding failure, this paradigm was buttressed with notions of thesaurus classes of words, statistical association probabilities, and superficial syntactic structures. The paradigm still proved inadequate as shown by the conclusions of the recent ALPAC (1967) report and by a sharp criticism of language processors by Kasher (1966). In the meantime, Chomsky (1965) devised a paradigm for linguistic analysis that includes syntactic, semantic, and phonological components to account for the generation of natural language statements.

He says (p. 141): "The syntactic component consists of a base and a transformational component. The base in turn, consists of a categorial subcomponent and a lexicon. The base generates deep structures. A deep structure enters the semantic component and receives a semantic interpretation; it is mapped by the transformational rules into a surface structure, which is then given a phonetic interpretation by the rules of the phonological component." This theory can be interpreted to imply that the meaning of a sentence can be represented as a semantically interpreted deep structure—i.e. a formal data structure.

From computer science's preoccupation with formal programming languages and compilers, there emerged another paradigm. In this one the elements of a language are formally defined objects in a well-defined syntactic structure. Translation between two such languages is accomplished by a set of transformational rules or functions whose arguments are these structured objects. The set of transforming functions is seen to be the semantics of the system, and the meaning of a program is generally taken to be the effect of its operation.

The adoption and combination of these two new paradigms have resulted in a vigorous new generation of language processing systems characterized by sophisticated linguistic and logical processing of well-defined formal data structures. It is my purpose to examine several of these systems and draw conclusions concerning the state-of-the-art, its principles, its problems, and its prognosis for socially useful applications.

BACKGROUND. In 1965, the first generation of fifteen experimental question-answering systems was reviewed (Simmons 1965). These included a social-conversation machine, systems that translated from English into limited logical calculi, and programs that attempted to answer questions from English text. The survey concluded that important principles of language processing were being unearthed and that significant progress was being made toward the development of useful question-answering systems. This conclusion was criticized by Giuliano who

* Department of Computer Sciences. This paper was supported in part by the US Air Force Human Resources Research Laboratory, Training Research Division at Wright Patterson AFB, under Contract F33615-67-C-1473, toward the development of a natural language CAI System, through a subcontract from the Systems Development Corporation, Santa Monica, California.

took the contrary view that there was "... evidence mainly of motion with little real evidence of progress" (Giuliano 1965). Kasher (1966) critically reviewed several example systems to conclude that none met even minimal criteria for successful semantic analysis or logical inference capabilities. Continued efforts in the field indicate that Giuliano's view was unnecessarily pessimistic, and these efforts have gone far toward correcting the inadequacies that Kasher stressed. Most systems developed since that time have included formally described data structures, explicit semantic analysis procedures, and a significant degree of deductive capability.

2. Second Generation Systems

First generation systems were not only handicapped by the lack of adequate linguistic models but in addition were often written in low level languages such as FAP and IPL. Considerable impetus was gained from the appearance of such higher level languages as COMIT, LISP, SLIP, SNOBOL, and ALGOL, which were commonly available by 1964 and 1965. The additional leverage added by the accessibility of time-shared, interactive consoles greatly eased the task of programming and debugging complicated language processing programs, and as a consequence, numerous language processing systems, many of them variants on the question-answering theme, were constructed. A group of conversation machines typified by Weizenbaum's ELIZA and Colby's belief system simulations were experimented with in several settings. A number of excellent approaches to natural language-oriented fact retrieval systems and a calculus word-problem-solver were programmed, and efforts continued toward the development of natural text question-answering and paraphrasing systems. For convenience of presentation, the second generation systems will be considered under the headings: conversation machines, fact-retrieval systems, mathematical word-problem-solvers, and natural language text processing.

CONVERSATION MACHINES. ELIZA is a SLIP program developed by Weizenbaum (1966) to explore the possibility of programming computers to conduct natural language conversations with humans. Early experiments with ELIZA simulated (or caricatured?) the conversational mode of a Rogerian psychotherapist as exemplified in the following brief excerpt from a computer conversation:

```
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE?
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED.
It's true, I am unhappy.
DO YOU THINK COMING HERE WILL HELP YOU NOT TO
  BE UNHAPPY?
Etc.
```

The program structure that supports this conversational capability is a set of pattern-operation transformations whose patterns are composed of keywords and whose operations are the substitution of a partially composed English statement in conjunction with some portion of the input sentence. These transformations are provided to ELIZA by a prepared script. The script is a high level program[1] whose statements are not necessarily sequential—the flow of control among command statements in the script is guided by the keywords of the input.

If we consider a brief example and its script rule, the central operation of the system can be seen.

```
Input: you are very helpful
Script: (0 you are 0) = (What makes you think I am 4)
Output: What makes you think I am very helpful?
```

In the input phase, which includes a dictionary lookup, the "you" elicits an associated script rule that matches the text. The pattern (0 you are 0) matches any occurrence of the sequence of words "you are" regardless of what preceded or followed them. At this point ELIZA uses the pattern to segment and number the modified input text as follows:

1: (empty); 2: you; 3: are; 4: very helpful

It then rewrites for the number 4 in the right half of the rule, "very helpful," the element labeled 4 by the left half, and prints the result: "What makes you think I am very helpful?"

ELIZA has been studied in application to a number of computer-aided instruction problems (Taylor 1968) and to a computer simulation of counseling behavior (Ellis and Tiedeman, 1968) and in each case has shown promising capabilities. The system includes limited features for remembering previously mentioned topics, for tagging keywords in the dictionary, and for operating script rules that refer to the tags.

Colby's simulations of belief systems originally used a set of pattern-operation rules much like those of ELIZA but have since progressed to a more complex organization (Colby and Enea, 1967; Tesler, Enea, and Colby, 1968; Colby and Enea, 1968). In original phases of this research, Colby's program scanned an input statement to discover particular patterns of words. When such a pattern was found, a response associated with the pattern was selected by the program and typed as a conversational reply to the subject's input. The purpose of the conversation was to allow the program to accumulate belief statements held by the subject. Augmentation of this direct word-pattern matching by the addition of pushdown stacks that helped to maintain the continuity of conversations supported Colby's hypothesis that a successful man-machine dialogue system required models of the patient, of itself, and of the world in general. The pushdown stacks in early implementations predicted certain aspects of replies to the computer system's questions; thus a "why" question predicted a "because" type response, and a "yes-no" type question predicted an affirmative or negative reply. Additional information stacks were used to remember pertinent facts known about the subject so that blunders such as the one illustrated in the following dialogue between the person and the program could be avoided.

[1] Similar in structure to parts of the pattern-operation rules of COMIT, METEOR, or SNOBOL.

When my father died my mother was sick for a long time.
TELL ME MORE ABOUT HIM.
My father was a writer. He was kind but rather cynical . . .
IS HE NOW?
He is dead.

The blunder occurred because the system at that time did not store and use the fact that the father was dead.

In the most recent phases of this research a directed graph model of concepts in relation to other concepts has been developed to model the belief systems of the persons being conversed with. Concepts are represented by nodes, and relationships by the labeled connections between nodes. Each node in the graph is associated with a set of attributes that can characterize a belief in terms of its intensity, credibility, etc. The relations that connect concepts are characterized by formal properties that can be used by the model in processes of interpretation and reasoning. Rules of inference are also embedded in this structure in the form of conditional statements with variables. (This type of rule is explained and illustrated in the section on Fact-Retrieval Systems.)

Two 1969 reports (Colby et al.) offer some detail on the system's present methods for evaluating beliefs. The principal inference rule that has been used is of the general form "A implies B", where "implies" is understood to mean a psychological expectation and the A term may involve multiple conditions. An essentially unlimited series of such rules can be applied as follows:

$$A \rightarrow B \quad B \rightarrow C \quad C \rightarrow D \quad \text{etc.}$$

Such a procedure gives inference power limited only by search time and by the validity of the "implies" relation. Discovering beliefs relevant to a given proposition is accomplished by replacing terms in the proposition by similar, opposite, or complementary terms and applying inference rules to obtain the set of beliefs related to that proposition. Beliefs are evaluated by a technique that computes credibility as a weighted function of consistency and evidential foundation of each belief. The resulting evaluations are used in heuristics that control the depth of search for relevant beliefs. Results with this system show that it is effective for retrieving relevant beliefs—although the cost in computing time is not cited.

Schank and Tesler (1969) describe a semantic system for eventual use in conjunction with Colby's belief model. This system is called a conceptual parser. It is based on a dependency grammar in which each syntactic constituent is tested against a conceptual semantics data base that shows semantic compositions consistent with knowledge of relations that hold in "the real world." The parsing system includes transformations in the form of pattern-operation rules so that it can extricate embedded sentences as in the following example:

John saw Texas flying to California.

$$\begin{array}{ll} \text{Parse:} & \text{John} \overset{P}{\Leftrightarrow} \text{see} \leftarrow \text{Texas} \\ & \qquad \uparrow P \\ & \text{John} \Leftrightarrow \text{fly} \overset{\text{to}}{\Leftarrow} \text{California} \end{array}$$

The symbol "⇔" refers to a subject predicate relation "←"to a direct object, "⇐" to a prepositional dependency, " ↑ " to a modifier, and the label "P" refers to past tense. The alternate interpretation that "Texas flies to California" is eliminated by the absence of the conjunction of "Texas" and "fly" in the conceptual data base. Schank's system is unusual in basing its semantic analysis on a dependency grammar, but it appears to be as effective as any other system that makes a semantic test following the formation of a syntactic constituent.

The belief system simulations of Colby et al. compose a very active project and one that is significantly advancing capabilities for natural language conversations with computers. There is a balanced use of deductive and inductive forms of inference, and with the eventual incorporation of sophisticated syntactic and semantic analysis procedures this line of research may result in a significant psychological model and an effective language processor.

Abelson and Carroll (1965) have also reported on a computer model of belief systems that is based on a network of logical relations between concepts. It uses an inductive logic to substantiate or reject statements in accordance with its beliefs. This logic is based on the notion of frequency of class instantiation. For example, given the statement:

"Left-wingers mistreat US friends abroad"

the system uses the inductive rule that the sentence will be considered credible if at least half the instances of the concept, "left-wingers," are connected in belief statements with one of the instances of the predicate, "mistreat US friends abroad." Thus, if "administration theorists" is one of two concepts that are considered to be instances of "left-wingers" and is connected to "coddle left-leaning neutrals," which is considered an instance of the first predicate, then the statement is accepted as credible. The system also includes deductive processes and models for rationalizing and denying its input statements.

A most interesting notion of complex meaning—the implicational molecule—is described in a more recent paper (Abelson and Reich 1969). Abelson holds that one of the more important aspects of language usage is pragmatic analysis, i.e. the production of plausible implications from sentences. The implicational molecule is a first approach to pragmatic analysis. Abelson illustrates with three responses to the sentence "I went to three drugstores." "A syntactically based system might respond 'How did you go to three drugstores?' A semantically based program might respond, 'What useful things did you buy in three drugstores?' But a pragmatically based program ought to be clever enough to ask 'How come the first two drugstores didn't have what you wanted?' "

An implicational molecule is a set of sentence classes bound together by psychological implication. Thus a sentence is a propositional element, and implication links bind such elements into a molecule. The set {A does X, X causes Y, A wants Y} is an example of an implicational molecule. Abelson postulates a *completion tendency* as a characteristic of human users of language. So, given some elements of the set, the others may be inferred. Thus, if it is given that A does X and X causes Y, there is some likelihood of inferring that A wants Y; it is even plausible, given only A does X, to infer that X causes some Y that A wants. Implicational molecules are named; the above example is called PURPOSE (Y, A, X) and the program is allowed to infer that A did X with purpose Y. The idea of implicational molecules is an important addition to a program's capability for conducting connected discourse with humans for filling in unexpressed premises, intentions, purposes, etc.

Abelson's program is stocked with a data base containing beliefs representing "the extreme right wing point of view of a well-known ex-ex-senator." He reports that the use of implicational molecules greatly enhanced the program's capability to simulate actual replies of the senator to input statements. The system, programmed in SNOBOL3 and limited to fixed format sentences for input and output, is a most suggestive model of internal symbolic processes that may intervene between statement and response in a conversational system. For language-processing researchers, both Abelson's and Colby's systems demonstrate methods for using inductive inference in question-answering and conversational machines.

Becker (1969) has presented a detailed analysis of the notion of "analogy" in the context of a general data structure for representing semantic information derivable from English sentences. He suggests that an important aspect of understanding new information is the process of locating previously stored information analogous to it, and making predictions on the basis of this previous experience. Becker outlines a process for inductive inference on the basis of analogies, which is justifiably more complex than methods used by Colby and Abelson. Another approach that may prove useful for inductive inference is McCarthy's (1963) formal system for the Advice Taker that introduces a modal logic for inferring that CAN$(a\ b)$ and CAN$(b\ c)$ may imply CANACHULT$(a\ c)$, i.e. that a can ultimately achieve c.

FACT-RETRIEVAL SYSTEMS. Second generation systems include a natural language oriented fact-retrieval system by Elliott (1965) and a generalization of several previous approaches into one system, DEDUCOM, by Slagle (1965). Both of these systems used a formal—but English-like—language as input. Slagle's DEDUCtive COMmunicator was a LISP system that stored LISP expressions of data statements such as:

1. There are 5 fingers on a hand
2. There is one hand on an arm
3. There are 2 arms on a man

and inference rules in the form of conditional statements that included variables as in the following:

4. If there are m X's on a V and if there are n V's on a Y, then there are mn X's on a Y.

By substituting data statements for the variables in conditional expressions, DEDUCOM answers questions such as the following:

Question: How many fingers on a man?
Answer: 10.

DEDUCOM can be considered as a tour de force in LISP that explores the deductive power of inference rules in the form of conditionals with variables and transformations—another form of the pattern-operation rule. Such rules were first introduced to the question-answering scene by Black (1964), Raphael (1964), and Bobrow (1964).[2] Although this form of transformation is powerful enough to deduce an answer to a question, given appropriate data statements, its application to a large data structure is hopelessly expensive without the inclusion of an appropriate set of tree-pruning heuristics.

Elliott's system, in contrast, operated very rapidly because of his careful attention to the development of efficient data structures. Input to this system is in the form of parenthesized natural English statements such as the following:

1 (Fact(San Francisco) (is north of) (Mexico City))

The canonical form for an input is:

(Operator(Datum$_1$) (Relation) (Datum$_2$)).

The system builds a directed graph to represent the relations between its data terms. A relation is defined to the system by a set of properties such as reflexive and symmetric. The pattern of properties associated with a relation is used by the system to call a subroutine that constructs appropriate connections between the data terms.

This system also uses conditional rules with variables, as in the following form:

(Combine ((B) (is between) ((A) and (C))) IF ((B) (is less than) (A)) AND ((C) (is less than) (B)))

The use of these conditional transforms gives the system great deductive power, and since it depends mainly on a strongly ordered data structure, it is able to operate rapidly in answering most queries.

Neither Slagle nor Elliott chose to confront the problem of syntactic and semantic analysis of English statements and queries but instead explored the implications of their respective structural models in terms of deductive power and retrieval effectiveness. Both emphasize the deductive power of conditional transforms with variables, but Elliott introduced the effective idea of characterizing relations by properties which are used in ordering the data in a directed graph.

Since these 1965 programs, several small scale natural language processors and several very large formal language data management systems have been developed. These are cited briefly in the section headed Miscellaneous. Of more significance for this review are three recent fact-retrieval systems, each of which confronts the semantic problems of English and includes a deductive capability

[2] These and other relevant language processing theses and papers have recently been collected as a book by Minsky (1969).

for answering questions (as contrasted with the direct lookup operations of the formal language systems).

The first of these is a merger of Raphael's earlier line of thought, first with formal theorem-proving techniques (Green and Raphael 1968), then with a natural language semantic system developed by Coles (1968). Coles' approach to linguistic analysis includes a one-pass predictive syntax recognizer or parser that is automatically produced from a BNF description of the grammar via an algorithm developed by Earley (1965). It may be that such an approach to compiling a parser optimizes its efficiency with respect to a subset of English, but it appears to entail the disadvantage of requiring a new recognizer to be compiled each time the grammar is changed.

The semantic approach taken by Coles includes the notion of a model of what is being talked about and an unambiguous formal language, the predicate calculus, which can express the facts of the model. His semantic analyzer must transform constituents of an English statement about the model into constituents of predicate calculus statements. Disambiguation is achieved by testing the truth value of the logical statement in terms of the model. These formidable tasks are accomplished with the aid of production rules or transformations associated with each syntactic rule. Coles cites earlier uses of this technique by Kirsch (1964) and a similar principle used by Thompson (1964) as the basis for semantic systems. A recent paper by Kochen (1969)[3] reports the detailed design of a similar system that uses production rules for transforming from English into predicate calculus statements and questions about simple diagrams as a stage in the production of flowcharts of programs for answering the questions. Because the principle of transforming natural language constituents into the formal language of the system is now commonly used in second generation language processors, it is illustrated here in some detail.

Suppose that the sentence "Each resistor is an element." is to be represented in the predicate calculus as "$(\forall x)[\text{resistor}(x) \Rightarrow \text{element}(x)]$." The following grammar serves:

$$\begin{array}{ll}
S \rightarrow NP_1 + PRED & \mapsto \text{“}(\forall x)\,[\alpha(x) \Rightarrow \beta(x)]\text{”} \\
NP_1 \rightarrow DET_1 + N & \mapsto \text{“}\alpha(x) \leftarrow N(x)\text{”} \\
PRED \rightarrow V + NP_2 & \\
NP_2 \rightarrow DET_2 + N & \mapsto \text{“}\beta(x) \leftarrow N(x)\text{”} \\
DET_1 \rightarrow \text{EACH, EVERY} & \\
DET_2 \rightarrow \text{A, AN} & \\
N \rightarrow \text{RESISTOR, ELEMENT} & \\
V \rightarrow \text{IS} &
\end{array}$$

The operation of this grammar is similar to that of an ordinary context-free phrase structure grammar except that as each rewrite rule is successfully applied, the associated semantic transformation is executed on a separate semantic pushdown list. Thus, with a top-down approach, the symbol S is selected and rewritten as $NP_1 + PRED$; NP_1 is then rewritten as $DET_1 + N$; DET_1 is found in the input sentence to be the initial word, "Each"; PRED is stored on the syntactic pushdown list; N matches the following word "resistor" so the semantic transformation associated with NP_1 now can be applied to the semantic pushdown list giving us

$$\alpha(x) \leftarrow \text{resistor}\ (x),$$

while the input string is rewritten as

$$NP_1 \text{ is an element.}$$

Returning to the syntactic pushdown list of goals, the term PRED is then rewritten as $V + NP_2$; V is found to match "is" with no semantic transformation required; NP_2 rewrites as $DET_2 + N$, which matches the words "an" and "element" respectively. Now, according to the semantic transformation associated with NP_2, we obtain a semantic pushdown list of

$$\beta(x) \leftarrow \text{element}(x), \quad \alpha(x) \leftarrow \text{resistor}(x)$$

and have an input string of

$$NP_1\ V\ NP_2.$$

Finally, the semantic transformation in the rule for S gives us

$$(\forall x)[\alpha(x) \Rightarrow \beta(x)], \quad \beta(x) \leftarrow \text{element}(x), \quad \alpha(x) \leftarrow \text{resistor}(x)$$

which upon evaluation results in the desired form

$$(\forall x)[\text{resistor}(x) \Rightarrow \text{element}(x)].$$

Coles' variation of this technique is sufficiently strong to translate certain English sentences into a fully quantified predicate calculus—provided that his grammar and transformations are sufficiently detailed to recognize the subtle cues that distinguish various meanings of "each," "every," "the," etc., in their quantificational function and the sometimes even more subtle cues that signify the scope of the quantifier. One strategy that Coles has not fully capitalized upon is to test the semantic well-formedness of each major constituent as it is constructed. This has been found in other studies (below) to be an important pruning heuristic for reducing the number of meaningless constituents that are carried during the analysis.

Green and Raphael's system for answering questions deductively uses the Robinson resolution theorem proving procedure which finds proofs by refutation. The question is taken as a postulated theorem. The resolution procedure then attempts to construct a model that satisfies both the axioms and the *negation* of the theorem; i.e., if the theorem does follow from the axioms, then such a model does not exist, and the resolution procedure discovers this by deriving a contradiction in its attempt to construct the model. If the theorem is not proved after some given effort is expended, an attempt is made to show that the theorem is false by assuming it true and searching for a contradiction. The process continues until either a proof or a disproof is found, or until some allotted amount of search time is exceeded. With the aid of heuristics to deal with the more closely related axioms first, and to avoid repeating equivalent proofs, the researchers believe that the approach may develop into one of practical usefulness on data bases of reasonable size. The 1969 papers quote a number of difficult example questions that were successfully answered by the system,

[3] This paper publishes research first reported by Kochen in 1965.

QA3, using this technique, and show its application to general problem solving and to commanding actions from the SRI robot (Green 1968, Coles 1969).

The resolution method was first experimented with in a natural language application by Darlington (1965), who has since, in two reports (1969), contributed further refinements and additional experiments with the method's effectiveness in a near-English formal language question-answering system. Proponents of this approach to deductive question answering and automated theorem proving generally follow Robinson's (1967) enthusiastic conclusion (in reference to mathematical contexts): "A theorem proving problem can be solved automatically if it can be solved at all . . . by executing a certain purely clerical algorithm. . . ." At the same time, those concerned with question-answering systems have encountered well-known shortcomings in the first order predicate calculus, with reference to representing equality, recursiveness, modality, tense, and the detailed quantificational structure of English. In response to shortcomings of first order logics, Robinson (1968) attempts ". . . to persuade those engaged in mechanical theorem-proving research, and those proposing to start such research, to focus their attention henceforth on mechanizing higher order logic."

In actual fact, researchers are quietly extending the first order predicate calculus by introducing additional operators and quantifiers. (See for example, Green and Raphael's *var* operator and Woods' use of the *iota* quantifier and successor functions.) John McCarthy has contributed a useful formulation of a modal predicate calculus for dealing with the notion of "can" in his Advice Taker, and Pople (1969) has extended these notions somewhat in his goal-oriented language (GOL) for the general problem-solving area.

As a culmination of several years of research on data management systems Kellogg (1968) has developed a system for compiling formal language data management procedures from a subset of natural English statements and questions. Unique to the Kellogg system is a satisfying sense of completeness; first, it is programmed and operating as a complete system; second, it accepts natural language questions and statements and retrieves data or modifies the data base; third, it includes minimally adequate syntactic and semantic analysis approaches that are based on current linguistic theory; finally, it incorporates sufficient logical structure to support deductive procedures based on both mathematical and logical relations. A significant weakness is that, as an experimental system, it is currently limited to operation on data bases that can be contained in core memory; however, the present line of research is aimed at expanding the approach to auxiliary storage.

Kellogg defines a formal language for information management. He requires that such a formal language: (1) be procedural, machine independent and independent of special data requirements or considerations; (2) approach the power of the predicate calculus in its capabilities for composition of functions, nesting of relations, embedding of procedures within procedures and representing quantification over sets; (3) be easy to read and understand. The language he defines in 30 (complex) formation rules is shown by numerous examples to be adequate for expressing complex data retrieval requests and for describing data for storage. His implementation of the quantificational feature is still limited.

The linguistic procedure for translating from an English string into the formal language structure begins with a top-down syntactic analysis based on a context free phrase structure grammar. The lexical structure associated with each English word includes syntactic and semantic word-classes, a list of semantic features, and a list of selection restrictions. As the syntactic parser constructs a constituent, a semantic test is made to discover if the features of the head of the construction satisfy the selection restrictions of the dependent construction. If the test is satisfied, a transformation is effected to compose (i.e. combine) the features and selection restrictions for the resulting constituent. The semantic test and composition functions follow closely the notions outlined by Katz (1967) but are more explicit than the limited descriptions offered by Katz.

Following the semantic composition of a constituent, additional transformations may be signaled to translate it into a portion of the resultant formal language expression. The process, with the exception of the semantic composition functions, can be seen to follow roughly the example previously illustrated by the sentence "Each resistor is an element." Once again we see the application of the powerful pattern-operation rule, this time for disambiguation by use of semantic features and selection restrictions as well as for transformation into a formal language expression that is operable as a program applied to a data base.

The third natural language data base system, designed by Woods (1967, 1968), begins by analyzing the data from an airlines guide into a set of primitive functions and predicates. Predicates include such examples as the following:

CONNECT (X1, X2, X3)	Flight X1 goes from place X2 to place X3.
DEPART (X1, X2)	Flight X1 leaves place X2.
MEALSERV (X1, X2)	Flight X1 has type X2 meal service.
PLACE (X1)	X1 is a place.
FLIGHT (X1)	X1 is a flight.

Examples of primitive functions include the following:

DTIME (X1, X2)	Departure time of flight X1 from place X2
OWNER (X1)	Name of airline that operates flight X1
TZ (X1)	Time zone of place X1

The primitive predicates and functions comprise the elementary operations of a procedural language for managing a data base of airline guide information. Each predicate may be tested as true or false and each function can be operated to return a value. The meanings of the primitives are thus defined by programmed subroutines which may be combined into more complex programs to define additional predicates and functions. The procedural language also includes the data management operators, RECORD, LIST, TEST, PRINT, etc., and a detailed expression of quantification, EACH, EVERY, ALL, SOME, etc., and its scope.

The language processing task is to translate from natural language statements or queries into a quantified formal expression in the procedural language. As input to his semantic system, Woods uses a deep structure syntactic analysis of each sentence in the form of a labeled phrase marker, like that output from the Harvard syntactic analysis system. His semantic system depends on the use of pattern-operation rules whose left halves are Boolean combinations of labeled fragments of phrase markers that include specification of their terminal elements as English words or as semantic class predicates.[4] (By semantic class predicate is meant such predicates as PLACE(Boston)—True: Boston is a kind of place; or AIRPORT(Boston)—True; etc.) The right half of these rules is a function or predicate in the procedural language with its arguments specified as elements from the syntactic subtrees of the left half. These transformations are very powerful in that they incorporate detailed specification of legitimate combinations of syntactic features and semantic class markers (i.e. predicates) as conditions of the formation of (i.e. transformation to) procedural language constituents. They thus include tests for syntactic and semantic well-formedness to allow for selection of sense-meanings or disambiguation of words and phrases.

At all levels of constituent processing, translation from English determiners and quantificational terms to quantifiers in the formal language is treated carefully and successfully. It is probably in his detailed description of a method for dealing with quantification that Woods makes his most significant contribution. In other systems large gaps are to be found in the explanation of this process.

Woods' approach shows how "meaning" can be operationally defined as a sequence of subroutines, functions, or operations that a statement calls for a system to perform—so far in the context of a data management task. The extent to which this approach can generalize beyond the data base context remains to be discovered as does its effectiveness in treatment of various subtleties of English usage—that are beyond the immediate goals of Woods' research. One apparent weakness in the approach as so far described lies in Woods' use of syntactically analyzed English as an input to the system. A syntactic parsing that is independent of the semantic operations implies that numerous syntactically valid but semantically impossible interpretations would have to be considered. As he continues his development of the system, Woods expects to adopt a procedure of semantically testing each syntactic constituent as it is formed to dispose of semantically invalid constituents at the earliest possible moment.[5]

Mathematical Word-Problem Processors. Intermediate between data base systems and research on text processing, Charniak's (1969) CARPS is a program that solves calculus word problems. CARPS is a generalization and expansion of techniques introduced by Bobrow (1964) in his STUDENT program for solving algebra word problems. These two systems take a heuristic approach to the analysis of English sentences using pattern-operation rules for both syntactic and semantic operations.

Charniak describes the operation of his system with reference to the following example problem. "Water is flowing into a conical filter at the rate of 15.0 cubic inches per second. If the radius of the base of the filter is 5.0 inches and the altitude is 10.0 inches, find the rate at which the water level is rising when the volume is 100.0 cubic inches."

The first step in analysis is a dictionary lookup during which words may be tagged with syntactic information, common phrases are transformed to a canonical form, and keywords are noted if they have equations associated with them in memory or if they give information about the type of problem (e.g. volume or distance). The next phase is to transform the complicated sentences of the problem into simpler form. This phase is accomplished by pattern-operation rules as shown below with reference to the second sentence of the problem. After dictionary lookup this sentence appears as:

(A) (If the radius of the base of the filter (is verb) 5.0 (inches unit) and the altitude (is verb) 10.0 (inches unit), (find Qword) (rate Rword) at which the water level (rising verb) when the volume (is verb) 100.0 (IN3 UNIT))

The first relevant pattern is as follows:

(B) IF-ANYTHING-, -QWORD-ANYTHING.

Pattern (B) successfully matches the first clause of sentence (A); its associated operation is to break the sentence into two sentences, the first containing "if the radius . . . and the altitude is 10.0 inches" and the second beginning "Find. . . ." The rule then starts the program over, and again applies rules to each of the sentences. After the input sentences have been simplified, additional pattern-operation rules are applied to transform each sentence into equations or data structures. For example:

(C) ((water NVP) (flowing verb) (at PREP) (rate anything) (15.0 Number) (IN3 UNIT) (Per PER) (Second TIMEUNIT))

is matched by the rule:

(D) ANYTHING-VERB-PREP-ANYTHING-NUMBER-UNIT-PER-TIMEUNIT.

Rule (D) classifies the sentence as one in which the noun

[4] Woods shows the correspondence of these structures to semantic markers and selection restrictions of Katz's theory.

[5] Personal communication with Woods.

phrase (NVP), is changing at a constant rate and additional patterns and operations are applied to give the structure

```
Water
  Volume:  G0015
  Value:   (QUOTIENT(TIMES 15.0(TIMES TIM
           (EXPT IN 3)))SEC)
```

which includes the Lisp equivalent of the formula

$$\frac{\text{TIME} \times \text{INCHES}^3 \times 15}{\text{SECONDS}}$$

The result of these operations is to produce a tree structure of the information contained in the problem as in Figure 1. Final phases of the program establish equations from the structured information and solve them.

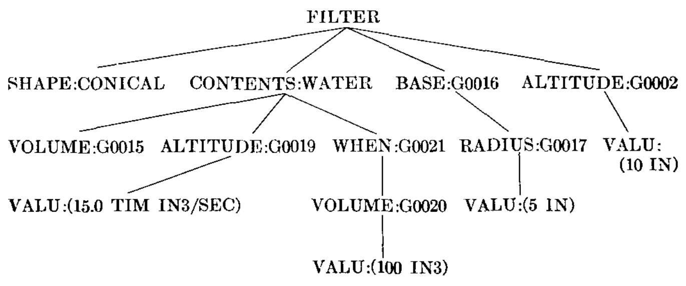


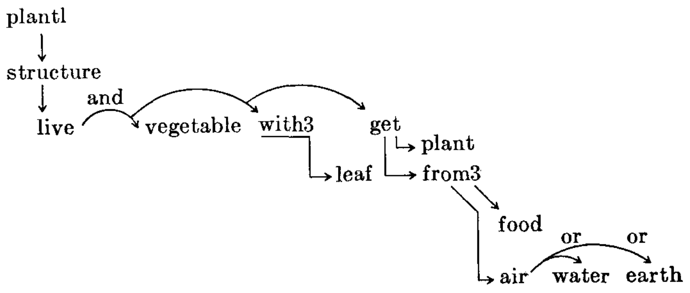
Fig. 1

Significant in this approach to language processing is the fact that paragraph units of natural (though specialized) text are dealt with. Problems of anaphora and pronominal reference are faced. For example when pronouns are encountered, a function returns the most likely referent. In addition, the accumulation of all information in the problem into a single tree structure provides a form of discourse analysis in which the phrase "the filter" in the second sentence of the problem refers to "the conical filter" already established in the data structure by the first sentence. Similarly "the altitude," "the water level," and "the volume" all refer back to attributes of the "conical filter."

Although Charniak's (and Bobrow's) systems effectively use pattern-operation rules to syntactically analyze their problems, Charniak concludes that an incremental left-to-right parse would be more efficient. We have previously seen syntactic constituents transformed into predicate calculus statements, Lisp functions, and other data structures with use of the pattern-operation rule; in these two programs the operations transform constituents into data structures and mathematical equations.

Natural Language Text Processing. The natural language fact retrieval systems just described define a subset of English and a formally defined data base structure whose content is usually a set of short fact statements. For text-processing applications, a major research task is to analyze a much broader subset of English into a data structure sufficiently general to represent something typical of the wide range of meanings expressed in a corpus of expository text. The natural model for this data structure is the invisible cognitive structure of the human language user. As a consequence, the analytic text-processing research has a pronounced psychological flavor as typified by Quillian's model of semantic memory structure (1967, 1968) and his recent Teachable Language Comprehender (1969), and Simmons' model of verbal understanding (Simmons et al. 1968). It is not surprising that these systems relate strongly to such other psychological models as Colby's and Abelson's simulations of belief structures in their mode of representing conceptual information.

Quillian's original semantic memory encodes dictionary definitions of words as a network of word-nodes connected by syntactic and semantic relations. The definition of an English word such as plant is characterized by a disjunctive set of labeled planes, e.g. plant1, plant2, plant3, etc., to represent the various alternate sense meanings. Within a plane the sense meaning is expressed by a structure such as:



Five types of relations are used to connect nodes in the graph. The first is the semantic relation of type or class illustrated by the connection of "plant" and "structure" by a single straight line with an arrow joining a type and token node. The syntactic notion of modification is symbolized by the same arrowed line connecting two tokens, as "structure" and "live". The relations of conjunction and disjunction are symbolized by labeled curved pointers. Finally, a relation signified by an English verb or preposition is shown by twin pointers to the subject and object of such a relational word. The notion of word type is reserved for the head of a sense meaning while a token is the representation of a word type as it is used in defining some other word type.

The result of this structure for each definition is a form of handmade syntactic and semantic analysis, the substructures of which tend to be similar in some respects to deep linguistic structures. Each plane represents the immediate definition of a concept; but a *full concept* is defined as all nodes that can be reached by an exhaustive tracing from the head word type of the plane. If we think of the semantic memory as a horizontal spider web, picking it up by any one node orders all other nodes in a vertical dimension with reference to the node selected. It is this vertical ordering of the web that Quillian defines as a full concept.

A primary operation on the network is to compare and contrast the meaning of any two word-concepts in the memory store and generate an English statement to represent the relationship. Thus, comparing "plant" and "live," the system reports:

1. Plant is a live structure.
2. Plant is structure which get food from air. This food is thing which being has-to take into itself to keep live.

In recent research Quillian (1969) has generalized his semantic network and embedded it in a program called the Teachable Language Comprehender (TLC). When new text is input to TLC, the system relates each assertion of the new text to its semantic memory which represents the facts it already has recorded. Comprehension of new terms is accomplished by a generalization process illustrated in the following example.

Suppose the program is faced with the new phrase, "client's lawyer." The two following facts are already stored in its memory: (1) a client is a *person* who employs professionals; (2) a lawyer is a *professional* who represents or advises a person in legal matters. Both of these facts are in the form of a superset class modified and distinguished by subproperties. By the intersection process described earlier, TLC finds that "client" and "lawyer" intersect in the node, "professional" which is a part of the property describing "client" and is the superset class for "lawyer." By letting "lawyer" substitute for the "professional" employed by "client," the system has identified a relationship that exists in the semantic network that may explicate the meaning of the input phrase. A syntactic test of the input is then made to determine if this relationship in memory corresponds to the form of the input; if the test succeeds, the new phrase has been correctly comprehended. In experiments with the above example, TLC demonstrates its understanding of the phrase "lawyer's client" by printing "under discussion is a client who employs a lawyer: this client is represented or advised by this lawyer in a legal matter."

A unique aspect of Quillian's approach is that all syntactic and semantic information is carried in a data base encoded as a network of interrelationships between words and word senses. His notion of a full concept as (essentially) all of a person's knowledge ordered with respect to that concept is a challenging one that adds richness to the idea of semantic analysis. In his view, disambiguation is to be accomplished generally by selecting those word-senses that have the shortest paths between them as the senses that are relevant to the context. This viewpoint is more flexible than Katz's calculus of selection restrictions and semantic markers and is one that can eventually account for the metaphorical usage of words quite as readily as it can for literal usage. However, beyond some limited explorations by Sparck-Jones (1965) there is practically no knowledge of the extent to which semantic distance measures will prove a successful technique for disambiguation.

The long continued line of synthex research has most recently resulted in Protosynthex III (Simmons et al. 1968, Schwarcz et al. (1968) that successfully analyzes a wide range of English sentences and questions, deductively answers many forms of question and generates English sentences either as answers to questions or as paraphrases of an input statement in English. At the base of this system is a model of human conceptual structures expressed as nested (Concept-Relation-Concept) triples. In this model a sentence such as "The angry pitcher struck the umpire who called the game" would be expressed by the following set of triples:

(((Pitcher MOD angry)TMOD the) (strike T past) (umpire SMOD(umpire(call T past) (game TMOD the))))

Each term in a triple is an unambiguous selection of a sense meaning of the word. Each middle term in a triple is a relation—not necessarily well defined. The structure is a formal language that expresses sentence meanings as a nested set of relational triples—i.e. binary relations.

Transforming from English to this formal language is accomplished by a bottom-up syntactic analysis using the Cocke algorithm.[6] Each constituent found acceptable in terms of the grammar is subjected first to a syntactic transformation, then to a semantic test. The grammar rules combine a phrase structure and a transformational component as in the following example:

ADJ NP — (B Mod A) NP.

Given the possible constituent formed of an adjective and a noun phrase, the rule applies. The transformation is found within the parentheses; in this case it takes from the left half, the Bth element, NP, the literal term Mod, and the Ath element, ADJ, to produce (NP Mod Adj) and the name NP. Sequences such as ABAAC are acceptable elements of the transformation to provide for reference to an element at any level of nesting.

Following the transformation the resulting constituent is tested semantically by looking it up in a list of rules called Semantic Event Forms or SEFs. An SEF is a triple of three semantic class terms. A semantic class is derived for a word, W, by testing it in the frame "W is a kind of ______." Thus pitcher is a kind of gameplayer; gameplayer is a kind of person; angry is a kind of emotion, etc. If the constituent in question were "angry pitcher" the result of the transformation gives "pitcher Mod angry.' The SEF rules include (person Mod emotion), and since person and emotion are semantic classes for "pitcher" and "angry," respectively, the semantic test is passed successfully for the person-sense of "pitcher" and fails for the sense "pitcher as a container."

After disambiguation via the SEFs and transformation into the formal language, a question, no matter how complex, is resolved into a nested set of simple questions—i.e. triples. Each of these is looked up in the accumulated data store. If direct lookup is unsuccessful for a triple, attempts are made to deduce the answer using deductive inference rules that are keyed either to properties of the relations in the system or directly keyed to the relational word-concept. Some of these rules are in the form of

[6] Described in Kay (1964).

program functions while others use the familiar pattern-operation form like:

((A sister B) and (B mother C) ⇒ (A aunt C))

Several forms of these rules have been analyzed into classes which can be expressed more succinctly than the above. Using Complex-Product as an operator, the above rule can be expressed:

((sister C/P mother) ⇒ aunt)

with some gain in program efficiency.

The system has been tested on a range of questions selected from those provided in Compton's Encyclopedia. The detailed description of the question-answering process by Schwarcz et al. (1968) shows several successful examples and an analysis of certain types of questions that are beyond its scope. The question "Who lost the Battle of Waterloo?" is successfully answered by the statement "Napoleon commanded the French Army which lost the Battle of Waterloo in 1815." The question "Does the monkey get the bananas?" followed by a series of verbal statements describing the situation is also successfully answered with the aid of appropriate inference rules. Answering "how" and "why" questions is not yet possible for the system.

Paraphrase is treated as a special limited form of a question and covered by the same logic. Generation of answers is accomplished by using an inverse of the grammar that accepts formal structure triples and transforms them into English strings that express their meaning.

Weaknesses in this system include inadequate treatment of quantification and a certain awkwardness of structure that results from the complex nesting of data statements. Its syntactic-semantic machinery has been tested on a wide variety of sentences—including some that are very long and complicated—and found to be very powerful. It is presently written in Lisp and corebound. It is also presently very slow. An expanded version that can use disk storage is now being programmed. The syntactic-semantic component of the new system is running—about fifty times faster than the original—and it is believed that the whole revised system will operate rapidly and effectively enough to test it thoroughly on fairly large bodies of text (i.e. 5 to 10 thousand words).

Protosynthex III, although experimental, is another system that offers a sense of completeness—this time as a general purpose language processor. It develops and is based on a psychological model of cognitive structure that is grounded in linguistic and logical theory. It demonstrates that very sophisticated language processing operations are well within the range of today's computing technology—though so far only for small subsets of the language.

Miscellaneous. Several additional systems beyond those reviewed above have been developed (or further developed) in the past few years. Many of them deserve detailed treatment, but to keep this review manageable in size they are mentioned here briefly.

A recent system programmed in Snobol3 by Shapiro extends the ideas developed by Elliott. (Shapiro and Woodmansee 1969). It allows more freedom in the definition of relations and, in contrast to Elliott's GRAIS, it remembers new relations after they have been defined. Shapiro's system avoids problems of English syntax and semantics by using a near-English input-output language.

Salton's SMART system has achieved a high level of development as a general purpose document and text retrieval system based primarily on statistical treatment of words in text (Salton 1968). His use of statistical phrase-matching techniques and his approach to developing and using thesauri are noteworthy advances in the information retrieval area. Thompson's now classic DEACON system was carried somewhat further after he left TEMPO but eventually abandoned for lack of research funds. Most recently Thompson has developed the REL system of on-line multi-access consoles, which among other powerful capabilities allows a user to define a subset of English as a query and response language for data management tasks. As of this writing there are no published descriptions of this system.

Tharp and Krulee (1969) describe a still incomplete system to answer English questions from analyzed text. Their approach parses text with a transformational analysis system developed by Petrick (1965). They transform the analyzed text into n-place relational predicates where the verb signifies a relation between the subject and object of the sentence. Conjunctions are relations whose arguments are the conjoined sentences. The resulting predicates are ordered with respect to the adjudged priority of the relational term for accessibility to a retrieval algorithm. Because of limitations of the grammar, the input text is first edited by hand. They propose to incorporate deductive question-answering procedures of the type now common in question-answering systems.

Formal language data base systems are currently receiving much publicity but typically offer little in the way of deductive logic or semantic techniques. One exception in this area is the excellent approach of Levien and Maron (1969). This system expresses information related to a large document collection in the form of relational triples and provides a user language that is a mixture of English and a simple symbolic logic. It includes deductive techniques that provide answers to such complex questions as "What is the organizational affiliation of all authors whose publications are classed as natural language question answering?" Despite attempts at efficient programming for use on a data base of 200,000 statements, retrieval times typically measured in minutes. (Levien's experience casts some doubt on the vaunted efficiency of other data base systems where retrieval times are not so frankly reported.) A group of researchers at Hughes Aircraft (Savitt et al. 1966) designed and developed an approach to a non-Von Neumann type computer based on an associative memory and a pattern-operation type of instruction code. They have most recently simulated the system on an IBM 360 computer. It is probable that a hardware version of this system would prove a great boon to the general area of symbolic processing including applications to information retrieval, data management, language processing, and artificial intelligence research.

On the natural language processing aspect, Rosenbaum (1968) in addition to providing a detailed transformational grammar for a subset of English has designed (1967) a grammar-based question-answering procedure that capitalizes on the power of transformational rules to show that linguistic deep structures can serve as a data base for a fact file. Schwarcz (1967) outlined a set of principles that serve as a sound basis for question answering. Wilks (1967) shows how pattern-operation rules can be used to produce a rough semantic analysis of the flow of content through a paragraph. Bohnert and Becker (1966) have continued Bohnert's line of research on transforming predicate calculus statements into English forms and have unearthed useful methods for dealing with the difficult problems offered by prepositional phrases, conjunctions, and comparatives. Klein (1968) has published a description of a system that simulates the behavior of a linguist as he develops a grammar and a morphology for a new foreign language. Improved approaches to syntactic analysis have been described by Bobrow and Fraser (1969), by Thorne et al. (1968), and by Martin Kay (1964).

3. Discussion

In this paper the implicit definition of a language processor has been a system that accepts natural language statements and questions as input, uses syntactic and semantic processes to transform them into a formal language, provides deductive and/or inductive procedures for such operations as answering questions, and generates English strings as answers. Most of the systems reviewed in this paper are incomplete with respect to one or more clauses of this definition, but taken as a whole it is apparent that the field has developed techniques for at least minimal management of each of these aspects of language processing. It will prove profitable to examine and summarize the methods now commonly used for syntactic and semantic analysis, the data structures used to represent content, inferential procedures for answering questions, and the approaches used to generate English statements as responses.

Syntactic Analysis. It was surprising to discover that most of the language processing systems depended on a top-down approach to syntactic analysis. In a recent review of parsing algorithms used in formal language compilers, Feldman and Gries (1968) reported that most of the compilers used a basic top-down approach but also used bottom-up techniques as pruning heuristics to prevent the parser from exploring all possible branches of the grammar. It appears that a bottom-up approach is necessarily more economical where a large grammar is involved as must eventually be the case for a natural language processor. The use of the input string to directly select the relevant subset of the grammar eliminates a great deal of exploration of irrelevant rules that all begin with such common elements as S, NP, PRED, etc., each of which must be followed to terminal expressions by the pure top-down system. In the worst but usual case (in natural language work) where all interpretations of a sentence must be found, the top-down approach essentially requires the abortive generation of all strings that are partially well formed with respect to the grammar.

Despite this criticism, syntactic analysis is accomplished effectively with reasonably efficient algorithms by the second generation systems. The grammars typically include a phrase structure component in combination with a transformational capability and, in most cases, can deal successfully with discontinuous constituents. Also, since the lexicon and grammar are clearly separated from the parsing algorithm, the systems generalize to a wide range of natural languages providing the linguistic data is available. With the exception of a few systems such as Kellogg's, the lexical component has received little attention and is used primarily as a means for associating syntactic word-classes to the vocabulary.

It is apparent that the second generation approach to syntactic analysis still generally ignores most of the syntactic subtleties of English including agreement, punctuation, pronoun reference, treatment of comparatives, etc. However, despite the rough and ready nature of the approach, it is quite clear that basic computational procedures are well understood for syntactic analysis, including the transformational component required to obtain various forms of deep linguistic or conceptual structure.

Semantic Analysis. With respect to semantics, the situation is encouraging but not yet well developed. A semantic analysis of an English statement is required at least to select the word-sense or senses appropriate to the context (i.e. disambiguate) and to transform the sentence into one or more expressions in an unambiguous formal language. Katz (1967) also includes the notion of a composition function that will express the meaning of any and every constituent as a combination of the meanings associated with the elements that comprise it. So far, only Kellogg's system uses composition functions in this sense.

The most satisfactory approaches to semantic analysis are seen in systems by Woods, Kellogg, and Simmons—and these leave much to be desired. Each of these systems uses something akin to Katz's semantic markers, but the markers are so far limited to the form of semantic classes and lack the extensive structure Katz now believes to be required in a marker. Kellogg also uses selection restrictions and composition functions that express a meaning of each constituent in terms of a combination of the markers and selection restrictions of the elements that comprise that constituent. Simmons uses semantic event forms which are essentially rules that show allowable combinations of semantic classes, and does not provide any explicit composition function. However, both Kellogg and Simmons use the economical procedure of taking each constituent as soon as it is found acceptable syntactically and testing it for semantic well-formedness. This approach minimizes the number of meaningless syntactic constituents that have to be carried during the parse.[7]

Woods' test for semantic well-formedness occurs after the assumed deep structure analysis. It is accomplished by testing the sequence of semantic classes and English words as being an acceptable left half of a semantic transformation rule. Coles' approach does not explicitly deal with semantic classes, although word-classes that form his grammar rules may in fact be such; and his test of a sequence of class categories as a left element in a transformation rule may thus be a semantic check. For Coles' system, the final check of semantic well-formedness is to test the resulting formal language translation against a model representing the true state of the relevant universe.

Perhaps the principle of accomplishing semantic analysis via pattern operation rules can be seen most clearly in Eliza, the simplest of the systems reviewed above. The semantic analysis of a word or a pattern of words for a computer is the selection of either a data structure or a pattern of operations that it signifies. In Eliza semantic analysis by pattern-operation is frankly simplistic—a pattern of keywords on the left and a formula for constructing a conversational response on the right. For Woods the keywords are structured in syntactic patterns and the operations are keyed by an ordering of subroutines on the right. Despite the complexity engendered by syntactic and semantic word-classes, markers, selection restrictions, etc., the same pattern-operation principle is what supports the semantic capability of the other systems. Disambiguation is accomplished by testing a segment of the English string as an acceptable sequence of semantic units. The semantic content is expressed as a formal language whose elements may be either data structures or procedures.

Quillian's approach to semantic analysis offers an initial exploration of one additional aspect, that of semantic distance between two terms each of which signifies a data structure (concept) in a connected network of such structures. In Quillian's system the semantic content of a constituent would be a combined data structure which included the concepts of each of its elements as well as all concepts on the shortest path between them. This notion offers the advantage of providing a computer definition of meaning that includes some of the associational richness of human uses of the term with respect to language.

It is clear that the pattern-operation rule is today serving as the key to semantic analysis. However, no one has yet experimented with more than the barest of literal content—the richness of natural languages in terms of metaphoric and connotational content is completely untouched.

Data Structures for Representing Content. In talking about structures to represent content there is, on the one hand, the linguistic notion of deep structure representation and, on the other, the very common question of convenient computer representation. Chomsky's deep linguistic structures serve the purpose of showing that the complexity of natural language sentences can best be explained as a matter of transformational combination of what are very like simple subject-predicate sentences in base structures. However, because of their linguistic richness, the Chomsky-type base structures are more suited for linguistic research than for computer representation of factual information. Several alternate forms of representation can preserve linguistic detail but offer structures that are more tractable to computation. Bohnert and Kirsch introduced the hypothesis that an appropriate deep structure representation of meaning is the predicate calculus. Several systems follow this notion. Fillmore (1967) has recently offered an attractive linguistic deep structure that resolves into a nesting of attribute value lists that are easily representable as computer structures (see Simmons 1968). Concept networks used by Colby, Abelson, Simmons, and Quillian all depend strongly on the notion of nested attribute value lists to form computer representations for units of meaning. The structure of relational triples to represent meanings as nested sets of binary relations has advantages also.

To the extent that the conceptual content of a natural language statement can be represented *both* as a convenient data structure and as a defined formal language, operations on some aspects of meaning become both computable and describable. The power and the limitations of such languages can be explored by mathematical and logical methods. Perhaps more important for such cases, the problem of computer "understanding" of natural languages can be seen (as noted by Kellogg and by Thompson 1966) to be a special, vastly complicated case of compiler design.

The systems that have been reviewed above almost invariably depend on associative storage of the syntactic and semantic information they use. To attain associative storage in a sequential computer, such systems as Lisp, Snobol, or Slip are usually used. The unfortunate consequence is that all the systems are currently corebound in random access cores that (after including operating and embedding systems) allow 20 to 40 thousand cells of storage. Serious uses of language processing require dictionaries of between 15 and 100 thousand entries, vast quantities of syntactic and semantic information and, eventually, storage of encyclopedias of text. If there is to be any hope for useful language processing systems on sequential computers, there must be a melding of the technology of managing truly large data bases on auxiliary storage with the necessarily associative processing of computational linguistics. Alternate solutions exist, of course, in the provision of 100 million word cores or the development of associative computers—neither alternative appears likely in the immediate future.

A final remark is required on the content of the data

[7] Schank's approach to semantic analysis is essentially similar to Simmons' (though independently derived) except in his use of a dependency grammar to form the syntactic constituents.

structure. Most effort has so far been spent in designing algorithms and computable structures that can conveniently contain the linguistic and factual content required by a language processor. Researchers are keenly aware that lying in wait for them is the gargantuan task of encoding tens of thousands of dictionary items and innumerable syntactic, semantic, and logical inference rules for accomplishing on a large scale those language processing tasks that experiment has shown possible with small subsets of a language. At this moment it is hardly possible to estimate the effort that will be required, but it is safe to assume that researchers will discover an entire new spectrum of problems deriving from the complexity that comes with size.

Inference in Question Answering. After a question and an answering text have both been translated into an explicitly structured formal language, the process of question answering can be seen to be essentially one of theorem proving. Several systems embody deductive approaches that use inference rules to expand and transform the formal expression of a question until it matches some combination of data structures. In these approaches, the ever-present pattern-operation rule with variables has been a key technique. Recently, the Robinson resolution algorithm used by Green and Raphael has shown itself to be an attractive deductive approach, and methods used by Woods and Kochen suggest the eventual automatic production of programs to accomplish inference. Initial uses of inductive approaches are also to be found, particularly in Colby's, Abelson's, and Becker's work, while McCarthy shows lines along which inductive logics may be formalized. Statistical induction techniques for question answering are used in Salton's SMART and Simmons' Protosynthex I (1964).

The last two systems automatically index large quantities of text and use keyword techniques for retrieving sentences, paragraphs, or articles relevant to a query. The matching of keywords is augmented by statistical controls on word-form and sequence to retrieve text passages whose words (or thesaurus classes) are most highly correlated with semantic elements of the query. It must be noted that these systems work effectively on large textual data bases with no effort required for providing grammars and sets of inference rules. As document or text retrieval aids, they already approach the practically useful stage, and develop a statistical correlation approach to inductive inference in question answering.

Since this statistical induction approach coupled with the effective use of automatically produced indexes of the text has proved so effective, it has earned legitimacy as a question-answering technique. Presumably, its eventual place in the scheme is to act as a first stage filter that selects from a large body of data that portion which is obviously relevant to a question. More refined deductive and inductive approaches can then be used on the resulting small—perhaps manageable—selection of relevant material.

Generating Coherent English. Only a few systems have been concerned with generating natural English responses. It is an essential feature of Eliza, a defined requirement on Protosynthex III, and a continuing concern for Klein's control of style. The process, as might be expected, is the inverse of the analysis of a natural language; but interestingly enough, it is not only a generation in the linguistic sense, it is also a translation—from formal language to English.

The ubiquitous pattern-operation rule is the key to this procedure also. Constituents of a data structure are assigned class names and used as the left half of the transformation. The set of constituents is transformed into other constituents or a segment of a natural language string by the right half of the rule, and the eventual output is the set of well formed English expressions permitted by the grammar (see Simmons et al. 1968). Such a system naturally generates all purely syntactic paraphrases for a given meaning structure. If the meaning structure is based on characteristics (such as semantic markers) whose patterns represent sense meanings of words, the rules for selecting a word to match the pattern will be very like grammar rules, and lexical paraphrase will result by producing all strings whose terminal elements contain the required characteristics. Little more exists, however, than initial experiments with the production of meaningful language statements. Klein's work has shown that the selection and control of stylistic restraints on the choice and placement of words is an area where much research can be profitably centered.

A Unifying Paradigm. Much has emerged from consideration of various language processing systems: the pattern-operation rule; the representation of content as operations or as structures; the use of deep linguistic or conceptual structure versus the use of predicate calculus types of representation; the idea of natural language compilation; the idea of translation from natural language to a formal language representation of meaning. Let us attempt a synthesis.

Four strands of thought can be seen: first, the linguistic paradigm of deep structures and transformations; second, the computer science approach to compilation, i.e. the transformation using symbol tables and production rules from problem-oriented languages to machine languages of operations and data; third, the psychological notion of cognitive structure composed of concepts and relations; and finally, the basic ideas of logical structure and inference as expressed primarily in the predicate calculus.

These strands of thought are deeply interwoven in the existing systems. For example, Kellogg uses a lexical structure deriving from Chomsky and Katz, with production rules familiar to syntax directed compilers, to transform from an English subset to a formal language resembling the predicate calculus. In each of the question-answering systems the question eventually becomes a formal language theorem whose validity is to be tested with respect to a set of manipulative axioms and accepted data theorems. Psychologically-oriented systems such as those of Abelson, Colby, Quillian, and Simmons are built around simulated or synthesized cognitive structures whose elements are concepts and relations, and which incorporate inference procedures as special cases of concepts. These systems also show a concern for inductive as well as deductive inference methods. Coles' and Kochen's approaches are clearly generalized from the treatment of programming languages, while Woods brilliantly extends Lisp ideas and Katzian semantics into a function and predicate-oriented natural language data management system.

I believe these apparently disparate approaches reveal a unifying paradigm in which the four strands are plaited into a single line of thought that can guide further language processing research. The basis of this paradigm is a structure to represent the relevant conceptual content (i.e. aspects of meaning) symbolized by a natural language string. This structure must be a formal language with the power of a higher order logic. Variables and operators in this language can be represented, respectively, as nodes and labeled connections in a directed graph which, with suitable associated information concerning belief values, etc., can be taken as a model of human cognitive structure. Statements in the formal language can be interpreted as procedures that modify the network, accomplish inferences, or do other work; or they can be treated as uninterpreted data structures. Two statements in this language are equivalent if there exist content preserving transformations that can convert one into the other or if, as procedures, they accomplish equivalent results. Establishment of equivalence is a theorem-proving operation where content-preserving transformations are the axioms, and known facts are established theorems; deductive or inductive procedures can thus be used to draw conclusions or to construct predictions.

I believe that such a formal language representation of the content of natural language statements corresponds in large measure to the semantically interpreted deep structures of Chomsky's linguistic theory, and that his theory, especially as interpreted by Joyce Friedman (1969), describes a suitable approach for generating natural language statements from the formal language representation. Our own experience is that the application of a set of linguistic transformations to constituents of statements in the formal language can produce surface structures in English syntax that generate natural English statements. Thematic control of coherence and connectivity between statements is still largely an unexplored area, but one that is opened to exploration in this framework.

For recognizing natural language strings, the paradigm calls for classes of elements, or combinations of these classes in the surface language string to be tested, first for syntactic well-formedness, then for semantic coherence with respect to context. If these tests are successfully passed by a natural language constituent, it is then transformed into a constituent of the formal language by means of a Chomsky-type transformation, a pattern-operation rule or, most generally, a function. Different natural languages are characterized by differing syntactic and semantic systems. But, since at least approximate translations between statements in natural languages are possible, it is reasonable to assume that the underlying cognitive structures of people who speak each language are structurally similar and can be represented by the same formal language. Translation between two natural languages can thus be seen to result from analysis of one language using its syntax and semantics into the base formal language, followed by a generation using the other language's semantics and syntax in terms of elements selected from its lexicon.[8]

Question-answering, conversational machines, and creative writing devices easily fit within the above paradigm, but each requires differing sorts of logical operations on the underlying formal language. The paradigm will easily generalize to applications that operate between modalities, i.e. picture-to-language, language-to-action, etc., using the formal language as an intermediary.

The nature of a paradigm in science is to yield to better ones, as it is gradually found to be incomplete or inadequate in the face of more finely articulated observations. It can be hoped that this one, emerging from the first decade of research in computational linguistics, will guide us well into the second decade before it becomes obviously obsolete. The second decade can be expected to test it on ever larger subsets of natural language materials in increasingly larger experimental contexts. We can hope that it will be extended to account for anaphoric, thematic, and discourse analyses of the paragraph and of larger units of natural language material.

4. Conclusions

In reviewing second generation question-answering systems it is apparent to me that significant progress has been made. Syntactic processing is well understood; semantic analysis has been operationally defined for small subsets of English, and certain limited, literal aspects of sentence meaning have been expressed as computable structure. The power of the pattern-operation rule, with or without variables, has been appreciated widely and exploited in application to semantic analysis and the deductive operations required for answering questions.

Significant weaknesses are still prominent. All existing systems are experimental in nature, small, and corebound. None uses more than a few hundred words of dictionary or a small grammar and semantic system. None can deal with more than a small subset of English strings. Deductive operations, though undeniably powerful, still generally lack adequate heuristic controls to limit the extent of searching an infinite inference tree. Little has been done so far to incorporate inductive inference procedures.

[8] This aspect of the paradigm was foreshadowed long ago by Yngve's (1957) analysis of the automatic translation problem.

Few systems (see Charniak and Wilks) go beyond sentence boundaries in their analyses, and generally acceptable methods for anaphoric analysis and the discovery of pronominal reference have not yet been developed. Such subtleties as the relativity of adjectives and adverbs, thematic sequence, metaphor, etc., have still to be explored.

In conclusion, minimally adequate methods have been developed for dealing with natural languages in small quantity. The next step, managing dictionaries with tens of thousands of entries and correspondingly large grammars and semantic systems will entail a whole new order of complexity and may require the invention of entirely new techniques to accomplish the same goals. I believe the next step requires confrontation of the problem of pure size. The time is almost upon us when we can consider a research program that proposes to exploit and explore existing techniques in application to a question-answering system based on a 20 to 40 million word encyclopedia.[9] Perhaps only with such a program can we expect to discover whether what has been learned so far can be used for an eventual practical question-answering system.

Acknowledgments. I am grateful to several colleagues for critical reading and editing of various drafts of this paper. Steve Coles offered a significant improvement to my halting description of the process of translating from English to the predicate calculus. Joe Becker (of Stanford University) found numerous errors, and his detailed comments suggested the line of thought from which the final unifying paradigm emerged. Dan Bobrow augmented my understanding of Quillian's recent efforts, and Bob Schwarcz has helped me comprehend linguistic and logical subtleties involved in many aspects of natural language processing.

Received January, 1969; revised June, 1969

REFERENCES

1. Abelson, R. P., and Carroll, J. D. Computer simulation of individual belief systems. *Amer. Behav. Sci. 9* (May 1965), 24–30.
2. ——, and Reich, C. M. Implicational molecules: a method for extracting meaning from input sentences. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 641–647.
3. ALPAC (Automated Language Processing Advisory Committee). Language and machines—computers in translation and linguistics. Nat. Acad. Sci., Washington, D. C., 1966.
4. Becker, J. D. The modeling of simple analogic and inductive processes in a semantic memory system. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 655–668.
5. Black, F. S. A deductive question-answering system. Ph.D. Th., Harvard U., Cambridge, Mass., June 1964.
6. Bobrow, D. G. A question-answering system for high school algebra word problems. Pro. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Pt. 1, Spartan Books, New York, pp. 591–614.
7. ——, and Fraser, B. An augmented state transition network analysis procedure. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 557–567.
8. Bohnert, H. G., and Becker, P. O. Automatic English-to logic translation in a simplified model. AFOSR 66-1727 (AD-637 227), IBM Thomas J. Watson Res. Center, Yorktown Heights, N. Y., Mar. 1966.
9. Charniak, E. Computer solution of calculus word problems. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 303–316.
10. Chomsky, N. *Aspects of the Theory of Syntax.* MIT Press, Cambridge, Mass., 1965.
11. Colby, Kenneth M., and Enea, Horace. Heuristic methods for computer understanding of natural language in context-restricted on-line dialogues. *Math. Biosciences 1* (1967), 1–25.
12. —— and ——. "Inductive Inference by Intelligent Machines." *Scientia, 103,* 669–20 (Jan.–Feb. 1968).
13. ——, Tesler, L., and Enea, H. Experiments with a search algorithm for the data base of a human belief structure. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 649–654.
14. —— and Smith, David C. Dialogues between humans and an artificial belief system. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 319–324.
15. Coles, L. Stephen. Talking with a robot in English. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 587–596.
16. ——. An on-line question-answering system with natural language and pictorial input. Proc. ACM 23rd Nat. Conf. 1968, Brandon Systems Press, Princeton, N. J., pp. 157–167.
17. Darlington, J. L. Machine methods for proving logical arguments expressed in English. *Mech. Transl. Comput. Linguist. 8* (June–Oct. 1965), pp. 41–67.
18. ——. Theorem proving and information retrieval. In *Machine Intelligence 4,* Meitzer and Mitchie (Eds.) Edinburgh U. Press, Edinburgh, 1969, Ch. 11.
19. ——. Theorem provers as questions answerers. Unfolioed mss. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, abstract p. 317.
20. Earley, J. C. Generating a recognizer for a BNF grammar. Comp. Ctr. Rep., Carnegie-Mellon U., Pittsburgh, Pa., June 1965.
21. Elliott, R. W. A model for a fact retrieval system. Ph.D. Th., TNN-42, Computation Center, U. Texas, Austin, Tex., May 1965.
22. Ellis, Allan B., and Tiedeman, David V. Can a machine counsel? Proc. CEED-SSRC Conference on Computer-based Instruction, U. of Texas, Austin, Texas, Oct. 1968.
23. Feldman, Jerome, and Gries, David. Translator writing systems. *Comm. ACM, 11,* 2 (Feb. 1968), 77–113.
24. Fillmore, C. J. The case for Case. In *Universals in Linguistic Theory.* E. Bach and T. Harms (Eds.), Holt, Rinehart and Winston, 1968.
25. Friedman, J. A computer system for transformational grammar. *Comm. ACM 12,* 6 (June 1969), 341–348.
26. Giuliano, V. E. Comments. *Comm. ACM 8,* 1 (Jan. 1965), 69–70.
27. Green, C. Cordell, and Raphael, Bertram. Research on intelligent question answering systems. Proc. ACM 23rd Nat. Conf., 1968, Brandon Systems Press, Princeton, N. J., pp. 169–181.
28. ——. Application of theorem proving to problem solving. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., pp. 219–239.
29. Hays, D. G. *Introduction to Computational Linguistics.* American Elsevier, New York, 1967.
30. Kasher, A. Data retrieval by computer: a critical survey. Hebrew U. of Jerusalem, Jan. 1966. Tech. Rep. No. 22 to Off. of Naval Res., Inf. Syst. Branch.

[9] Olney's careful preparation of computer usable versions of the entire contents of Merriam-Webster's Collegiate and Pocket dictionaries (Olney, *et al.* 1968) now completed, sets the stage for such an ambitious undertaking.

31. Katz, J. J. Recent issues in semantic theory. *Foundations of Language 3* (1967), 124–194.
32. Kay, M. A parsing program for computational grammars. RM-4283-PR, Rand Corp., Santa Monica, Calif., 1964.
33. Kellogg, Charles H. A natural language compiler for on-line data management. *AFIPS Conference Proceedings,* Vol. 33, Proc. AFIPS 1968 Fall Joint Comput. Conf. Vol. 33, Thompson Book Co., Washington, D. C., pp. 473–493.
34. Kirsch, R. A. Computer interpretation of English text and picture patterns. *IEEE Trans. EC* (Aug. 1964).
35. Klein, Sheldon, et al. The Autoling system. Tech. Rep. #43, Computer Sciences Dept., U. of Wisconsin, Madison, Wisc., Sept. 1968.
36. Kochen, Manfred. Automatic question-answering of English-like questions about simple diagrams. *J. ACM 16,* 1 (Jan. 1969), 26–48.
37. Kuhn, Thomas S. *Structure of Scientific Revolutions.* U. of Chicago Press, Chicago, Ill., 1962.
38. Levien, R. E., and Maron, M. E. Relational data file: a tool for mechanized inference execution and data retrieval. RM-4793-PR, Rand Corp., Santa Monica, Cal., Dec. 1965.
39. ——. Relational data file: experience with a system for propositional data storage and inference execution. RM-5947-PR, Rand Corp., Santa Monica, Calif., 1969.
40. McCarthy, J. Situations, actions and causal laws. Stanford Art. Intel. Proj., Memo 2, Stanford U., Stanford, Calif., 1963.
41. Minsky, M. *Semantic Information Processing.* MIT Press, Cambridge, Mass., 1969.
42. Newell, A., and Simon, H. A. GPS, a program that simulates human thought. In *Computers and Thought,* E. Feigenbaum and J. Feldman (Eds.), McGraw-Hill, New York, 1963.
43. Olney, J., Revard, C., and Ziff, P. Toward the development of computational aids for obtaining a formal semantic description of English. SP2766/001, Syst. Develop. Corp., Santa Monica, Calif., Oct. 1968.
44. Petrick, S. A recognition procedure for transformational grammars. Ph.D. Th., Dep. of Modern Languages, MIT, Cambridge, Mass., June 1965.
45. Pople, H. E., Jr. A goal-oriented language for the computer. Ph.D. Th., Systems and Communication Science Dep., Carnegie-Mellon U., Pittsburgh, Pa., 1969.
46. Quillian, M. Ross. Semantic memory. Ph.D. Th., Carnegie-Mellon U., Pittsburgh, Pa., Feb. 1966.
47. ——. Word concepts: a theory and simulation of some basic semantic capabilities. Doc. SP-2199, Syst. Develop. Corp., Santa Monica, Calif., Sept. 15, 1965. (Also in *Behav. Sci. 12,* (Sept. 1967), 410–430.)
48. ——. The teachable language comprehender. Bolt Baranek & Newman, Cambridge, Mass., Jan. 1969, in preparation.
49. ——. The teachable language comprehender. *Comm. ACM 12,* 8 (Aug. 1969), 459–476.
50. Raphael, B. SIR: a computer program for semantic information retrieval. Ph.D. Th., Math. Dep., MIT., Cambridge, Mass., June 1964. In Proc. AFIPS 1964 Fall Joint Comput. Conf. Vol. 26, Pt. 1, Spartan Books, New York, pp. 577–589.
51. Robinson, J. A. A review of automatic theorem proving. Proc. of Symposia in Applied Math, Vol. 19, AMS, Providence, R. I., 1967, pp. 1–18.
52. ——. The present state of mechanical theorem proving. Fourth System Symposium, Cleveland, O., Nov. 19–20, 1968, in preparation.
53. Rosenbaum, Peter. A grammar base question answering procedure. *Comm. ACM 10,* 10 (Oct. 1967), 630–635.
54. ——. English grammar II. Res. paper RC 2070, IBM Thomas J. Watson Res. Cent., Yorktown Heights, N.Y., Apr. 1968.
55. Salton, Gerard. *Automatic Information Organization and Retrieval.* McGraw-Hill, New York, 1968.
56. Savitt, D. A., Love, H. H., and Troop, R. E. Association-storing processor study. Tech. Rep. RADC-TR-66-174, Rome Air Develop. Center, Rome, N. Y., June 1966.
57. Schank, Roger C. Outline of a conceptual semantics for generation of coherent discourse. TRACOR-68-462-U, Tracor Inc., Austin, Tex., Mar. 1968.
58. ——, and Tesler, L. G. A conceptual parser for natural language. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 569–578.
59. Schwarcz, R. M. Steps toward a model of linguistic performance: a preliminary sketch. RM-5214-PR, The Rand Corp., Santa Monica, Calif., Jan. 1967.
60. ——, Burger, J. F., and Simmons, R. F. A deductive logic for answering English questions. Syst. Develop. Corp., Dec. 1968, in preparation.
61. ——. Towards a computational formalization of natural language semantics. *Proc. Int. Conf. on Comp. Ling.,* held in Sweden, Sept. 1969, in preparation.
62. Shapiro, S. C., and Woodmansee, G. H. A net structure based relational question answerer: description and examples. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 325–345.
63. Simmons, Robert F., Klein, S., and McConlogue, K. Indexing and dependency logic for answering English questions. *Amer. Doc. 15,* 3 (1964), 196–204.
64. ——. Answering English questions by computer: a survey. *Comm. ACM 8,* 1 (Jan. 1965), 53–70.
65. ——. Linguistic analysis of constructed responses in CAI. TNN-68, Comput. Center, U. Texas, Austin, Tex., Oct. 1968.
66. ——, Burger, J. F., and Schwarcz, R. M. A computational model of verbal understanding. Proc. AFIPS 1968 Fall Joint Comput. Conf., Vol. 33, Thompson Book Co., Washington, D. C., pp. 441–456.
67. Slagle, J. R. Experiments with a deductive question-answering program. *Comm. ACM 8,* 12 (Dec. 1965), 792–798.
68. Sparck-Jones, K. Experiments in semantic classification. *Mech. Transl. Comput. Linguist. 8,* 3 & 4 (1965), 97–112.
69. Taylor, Edwin F. (Ed.) ELIZA: a skimmable report on the ELIZA conversational tutoring system. Educational Res. Center, MIT, Cambridge, Mass., Mar. 1968.
70. Tesler, Lawrence, Enea, H., and Colby, K. M. A directed graph representation for computer simulation of belief systems. *Math. Biosciences 21,* 2 (Feb. 1968).
71. Tharp, A. L., and Krulee, G. K. Using relational operators to structure long-term memory. Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1969, pp. 579–586.
72. Thompson, F. B., *et al.* DEACON breadboard summary. RM64TMP-9, TEMPO, General Electric, Santa Barbara, Calif., Mar. 1964.
73. ——. English for the computer. Proc. AFIPS 1966 Fall Joint Comput. Conf., Vol. 29, Spartan Books, New York, pp. 349–356.
74. Thorne, J. P., Bratley, P., and Dewar, H. The syntactic analysis of English by machine. In *Machine Intelligence 3,* D. Michie (Ed.), American Elsevier, New York, 1968.
75. Weizenbaum, J. ELIZA—a computer program for the study of natural language communications between man and machine. *Comm. ACM 9,* 1 (Jan. 1966), 36–45.
76. Wilks, Yorick. Computable semantic derivations. SP 3017, Syst. Develop. Corp., Santa Monica, Calif., Jan. 1968.
77. Woods, William A. Semantic interpretation of English questions on a structured data base. Mathematical linguistics and automatic translation, Rep. NSF-17, 1967, Comput. Lab., Harvard U., Cambridge, Mass., Aug. 1966.
78. ——. Procedural semantics for a question-answering machine. Proc. AFIPS 1968 Fall Joint Comput. Conf., vol. 33, Thompson Book Co., Washington, D. C., pp. 457–471.
79. Yngve, V. H. A framework for syntactic translation. *Mech. Transl. Comput. Linguist. 4* (Dec. 1957), pp. 444–466.