DOCUMENT RESUME

ED 052 604 EM 009 055

AUTHOR Melton, T. R.

TITLE IT1: An Interpretive Tutor.

INSTITUTION Texas Univ., Austin. Computer-Assisted Instruction Lab.

SPONS AGENCY National Science Foundation, Washington, D.C.
REPORT NO TR-NL-4
PUB DATE May 71
NOTE 59p.

EDRS PRICE EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS *Computational Linguistics, *Computer Assisted Instruction, *Computer Programs, Generative Grammar, Language Instruction, Lexicology, *Program Descriptions, *Reading Instruction, Semantics, Structural Grammar, Syntax
IDENTIFIERS *Interpretive Tutor 1, IT1

ABSTRACT

A computer-assisted instruction system, called IT1 (Interpretive Tutor), is described which is intended to assist a student's efforts to learn the content of textual material and to evaluate his efforts toward that goal. The text is represented internally in the form of semantic networks with auxiliary structures which relate network nodes to particular sentences of the original text. The data base is completed with the addition of a lexicon, also in network form, which is used to store definitions and work-related syntactic information. A very simple command language is used by the student to request explanations, definitions, or quizzes over a specified set of sentences in the original text. A syntax-directed, natural-language sentence generator derives from the data base a set of simple sentences which preserve the meaning of the text passage specified by the student. The sentences may also be transformed into questions which help the student to evaluate his understanding of the text. A generative grammar and auxiliary functions were developed which produce sentences conforming to standard English practice with limited use of third-person pronouns. A conversational system was implemented in the LISP programing language. Potential applications of the existing system to language instruction are described. (Author)

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ED052604

IT1: AN INTERPRETIVE TUTOR

TECHNICAL REPORT NO. NL-4

T. R. Melton

May 1971

NATURAL LANGUAGE RESEARCH FOR COMPUTER-ASSISTED INSTRUCTION

Supported By:

THE NATIONAL SCIENCE FOUNDATION
Grant GJ 509 X

Department of Computer Sciences

and

Computer-Assisted Instruction Laboratory

The University of Texas
Austin, Texas

M 009 055

Abstract

A CAI system is described which is intended to be used to support the student's efforts to learn the content of textual material and to evaluate his own progress toward that goal. The text is represented internally in the form of semantic networks with auxiliary structures which relate network nodes to particular sentences of the original text. The data base is completed with the addition of a lexicon, also in network form, which is used to store definitions and word-related syntactic information. A very simple command language is used by the student to request explanations, word-definitions, or quizzes over a specified set of sentences in the original text. A syntax-directed natural-language sentence generator derives from the data base a set of simple sentences which preserve the meaning of the text passage specified by the student. These sentences may be transformed into questions by random selection of words to be omitted or replaced to produce, respectively, "fill-in-the-blank" or true-false questions.

A generative grammar and auxiliary functions were developed which produce sentences which conform to standard English practice with limited use of third-person pronouns. A conversational system was implemented in U.T. LISP 1.5.9. Potential applications of relatively simple extensions of the existing system to language instruction are described.

C O N T E N T S

LIST OF TABLES

LIST OF FIGURES

I N T R O D U C T I O N

Applying the term *Interpretive Tutor*, Simmons (1970) described a type of computer-assisted instruction (CAI) system which serves as a student aid in reviewing and promoting understanding of textual material. In order to accomplish this, the system includes the following components:

(1) *Discourse Structure:* a formalized representation of the content of the text.

(2) *Sentence Generator:* produces English sentences from a specified segment of the text by using the discourse structure as a data base.

(3) *Lexicon:* provides linguistic information to support the generation process and, also, contains definitions of the words in the text which may be displayed to the student on demand.

(4) *Tutorial Executive:* interprets student commands and controls communications between student and system. It also supervises execution of the tutorial functions and carries out the usual housekeeping activities required in CAI systems.

The interactions among student, instructor, course designer, and the Interpretive Tutor are illustrated in Figure 1. The student would be assigned a text to study which he would first read and then he would use the Interpretive Tutor to evaluate his retention of its content. If any of the text was difficult to read, the generative capacity of the system could be used to produce "explanations" consisting of simpler sentences paraphrasing the original text. The lexicon would provide definitions of words as used in the text. Quizzes might be generated by appropriate transformations of generated sentences. With these tools at his disposal the student could conduct a self-directed, self-motivated review of the text, repeating it until he had satisfied his own or instructor-imposed criteria of mastery. The successful

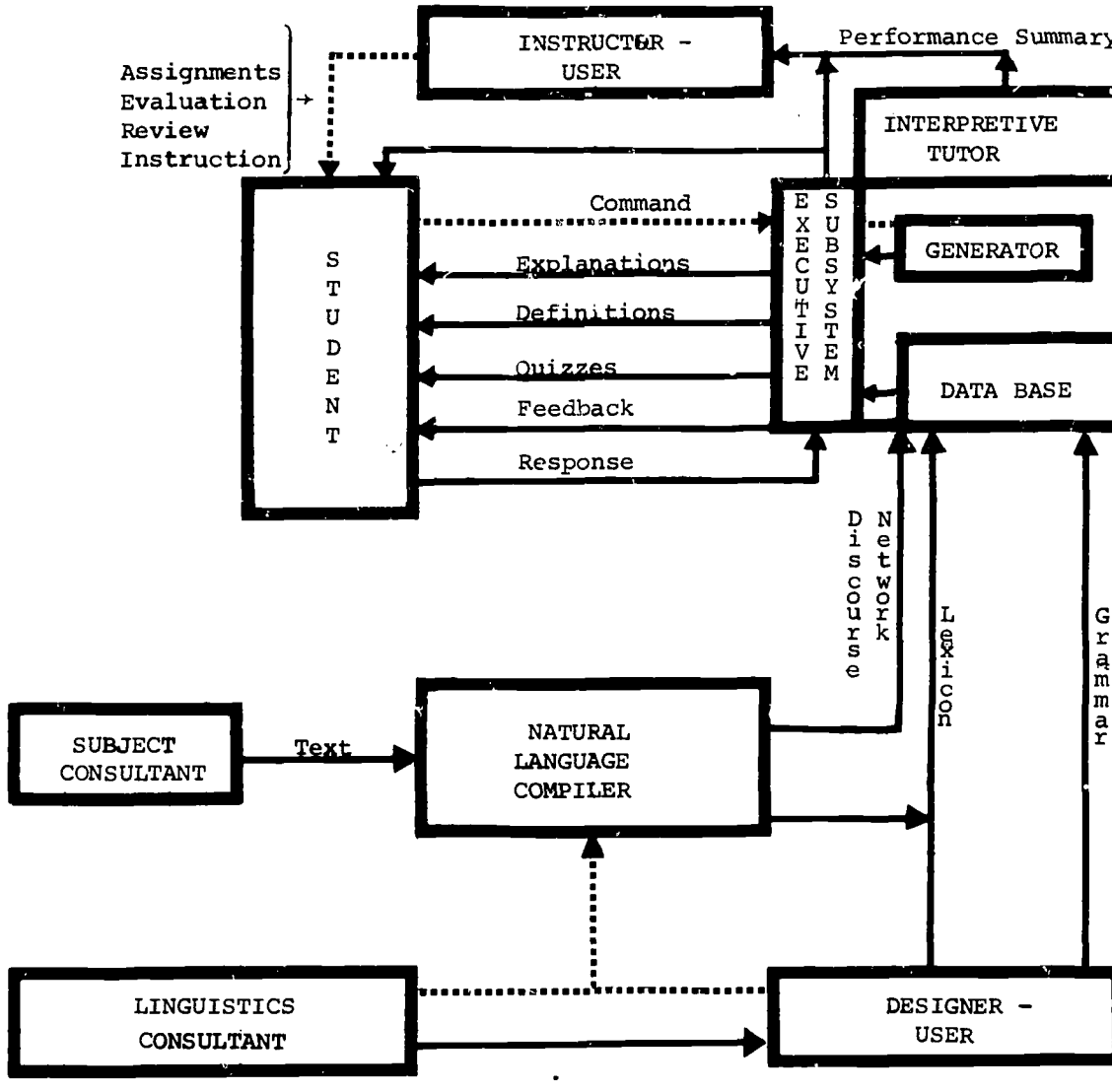


FIGURE 1. — Interpretive Tutor

implementation of such a system and its useful applications to CAI problems depend upon two important contributions from computational linguistics. The first is a mechanical aid to the preparation of the discourse structure. Manual preparation of these formal representations of text is a laborious and tedious clerical task. Simmons (1971) has developed a natural-language compiler which is capable of relieving human analysts of much of this labor. Yet, the analysis of natural language is not a task which can currently be entirely delegated to the computer, nor will it be for some time to come. However, an interactive version of this compiler is presently being developed. This will be used by the human analyst to perform the bulk of the analytical task, requiring only that he render aid in the translation of passages which are beyond the limitations of the automated system. It seems that the task of course preparation could be reduced to reasonable proportions by the use of such a system.

The second contribution from computational linguistics is a sentence generator of sufficient power to produce an acceptable variety of paraphrases. While existing systems are by no means capable of the expressive range of standard English, no such generative virtuosity is required for the task at hand. The purpose of the generative system is to simplify and clarify the text, rather than to produce impressive prose. Therefore, relatively simply syntactic constructions should serve the purpose of the Interpretive Tutor. Simmons and Slocum (1970) have developed one generator with sufficient power to produce acceptable sentences; another is described in this paper.

Complete fulfillment of the interpretive function requires a third contribution from computational linguistics and from artificial

intelligence. This would be a semantic component to permit interpretation of text or student responses. It would also generate questions and exercises in terms of the implications as well as the surface content of English sentences. Simmons (1968) experimented with a CAI system based on semantic analysis of student responses but concluded that a semantic component of sufficient generality could not be implemented practically today.

Full implementation of the Interpretive Tutor is a goal which, like all ideals, may never be reached; however, it may be approached step-by-step from any starting point. The purpose of this paper is to describe one step: an experimental CAI system, Interpretive Tutor 1, based on the technology of computational linguistics which might be practically applied today. The limitations of this initial system are subsequently discussed, after which some applications and simple extensions of the basic system will be proposed.

PURPOSE, GOALS, AND LIMITATIONS OF INTERPRETIVE TUTOR 1

Interpretive Tutor 1 (IT1) is a CAI system, but it is functionally based in computational linguistics. The experimental focus emphasizes the application of linguistic technology to instructional problems rather than the refinement of technology in either parent discipline. The purposes of IT1 are:

(1) To develop basic interactions between tutorial and linguistic components.

(2) To provide a foundation for estimating costs of implementation and operation of applied systems and for evaluating potential contributions of existing linguistic technology to educational problems.

To be useful as a basic instructional tool for a variety of topics, the structure of IT1 must admit considerable individual variety in discourse representation and syntactic expression in generated explanations and questions. This requirement was met by selecting a syntax-directed generator as well as a flexible structure for the discourse and lexical data base. These components are discussed later in detail.

IT1 is regarded as the experimental precursor of a family of similar but more relevant systems. A modular structure of program and data base is necessary to facilitate experimental modifications and extensions of the basic system.

The basic function of such Interpretive Tutor systems is to serve as a remedial and evaluatory tool in the general tutorial process. Thus they are not "tutorial CAI systems" in the sense of the term used by Zinn (1968). He referred to a tutorial system that controlled the

sequence of presentation of instructional material with a greater or lesser degree of individualized treatment according to the performance of a particular student. In applications of Interpretive Tutor, the human instructor and the student are expected to carry out the individualizing and frame-sequencing functions which consequently are omitted from the basic IT1 system.

Because of the experimental nature of IT1, several typical CAI functions have not yet been implemented. These include recording latency, responses, and scores for individual students and response-editing procedures for detecting misspelled words and similar types of trivial or mechanical student errors. Such features are essential when students are involved in either course development, evaluation, or educational operations. However, the examples of existing CAI systems provide an adequate basis for cost and design data, and the present experimental character of IT1 does not warrant their implementation.

The linguistic limitations of IT1 result more from the technological state of computational linguistics and economics than from restriction of purpose. A practical goal is sought. The means to that goal must not only be technically demonstrated, but the community of proposed users must also be able to afford those means. Simmons (1968) has concluded that, for economic if not technical reasons, the semantic analysis component must be set aside for the present. Without a semantic component, syntactic recognition of constructed responses is irrelevant. Hence, in the present system students are restricted to one-word responses.

The absence of the semantic component also imposes some restrictions on the performance of the sentence generator. It is clear that the

"explanations" generated by IT1 must be meaning-preserving paraphrases of the original text. It is neither necessary nor possible to reflect all human aspects of meaning in machine-generated sentences. Without attempting to define "meaning," the aspects of meaning which are of interest are illustrated by example and the meaning that IT1 can preserve in paraphrases is shown.

Consider the following family of paraphrases:

(1) The old man ate a fish.

(2) The old man did eat a fish.

(3) A fish was eaten by the old man.

(4) . . . the eating of a fish by the old man . . .

(5) The old man consumed a fish.

(6) (a) The old man took up pieces of a fish, chewed, and swallowed them.
(b) He continued in this way until the fish was gone.
(c) His belly then contained the masticated fish.

Sentence 1 is a straightforward statement describing an event in the universe external to the speaker and listener. Sentence 2 describes the same event, but the introduction of the emphatic modal "did" reveals some attitude of the speaker toward the event he describes or, perhaps, toward a skeptical listener. At any rate, the speaker's emphasis adds no information concerning the state of the universe external to the speaker-listener system.

Sentence 3 also describes the same event, but the speaker has chosen to emphasize the fish by choosing to mention it as the surface subject of the sentence. In 4, the entire event has been subsumed as a noun phrase which is embedded in a larger sentence. These choices may reveal an attitude of the speaker toward the event, or they may

have been influenced by previous subject choices in the larger context in which the sentence appears. In either case no information about the external universe is added to the content of the first sentence.

There is no equivalent in ITl to human attitude nor can ITl make a spontaneous choice among 1, 2, 3, or 4. However, it can produce any of these paraphrases if appropriate cues are stored in the discourse structure and if appropriate grammar rules are provided. The original subject choices are not identified in the discourse structure, and the author's attitude may be lost in generated sentences.

Sentences 1 through 4 might be regarded as purely "syntactic" paraphrases in that no content words were added to the standard description of the event which they all describe. Similarly, sentence 5 is almost a syntactic paraphrase since there is a sense of "consume" which is synonymous with the sense of "eat" used in the example. Given appropriate lexical data, ITl might produce sentence 5, but the risk of introducing spurious connotations seems to outweigh any possible benefit.

In contrast to sentences 1 through 5, sentences 6a, b, and c definitely comprise a "semantic" paraphrase. Whereas 1 through 5 present a face-value description of the event, 6 mentions several implications of it. Instead of simply declaring that the old man ate, 6a relates in some detail what it means to eat, and 6c mentions one of the usual consequences of eating. A fully interpretive system would permit the generation of 6a, b, and c under appropriate circumstances, but this is precisely the kind of operation which requires the semantic component which is not now available.

The generative power of IT1 is limited to syntactic paraphrases such as sentences 1 through 4. English permits many syntactic patterns within these limitations. If the language is defined to include sentences which are intelligible, without regard for conformation to any standard grammatical practice, the set of meaning-preserving paraphrases of an English sentence may be quite large. However, the basic purpose of IT1 requires that some minimal level of grammatical quality be observed. The informative function of IT1 would be compromised if the student were obliged to labor over "explanations" which did not conform to common usage and if the explanations did, by their very clumsiness, attract his attention to their syntax rather than to their content. Moreover, the systematic exposure of presumably impressionable students to substandard English usage is to be avoided.

The IT1 generator was consequently designed to conform to standard English usage at the level of simple sentences. Subject-verb number-agreement is enforced, nouns are inflected to indicate number and verbs for tense. Limited pronominalization is permitted. These features may be compromised in applications if it is necessary to reduce computational and storage requirements, but the basic capacity exists.

No extensive evaluation has been made of the ability of the present generator to produce compound or complex sentences. An experimental precursor of the present IT1 generator was reasonably successful in embedding subordinate clauses in complex sentences; however, that system was applied only to small discourse structures equivalent to two or three simple sentences. Generation of complex sentences from a large data base, such as that used in IT1, appears to be significantly governed

by a sort of super-grammar which is more closely related to rhetorical practice than to the syntactic principles that apply to single sentences. This rhetorical grammar is not adequately understood for general computational applications. As a result, the generative grammars used in ITl have been limited to production of simple sentences. This restriction does not seem to compromise the basic explanatory and question-generating purpose of the system.

OVERVIEW OF THE SYSTEM

Before describing ITl, it is appropriate to introduce the system by an example. The general process of application was given in broad outline in Figure 1. A subject expert prepares or selects a text to provide the data base. Each sentence in the text is given an identifying sentence-tag, such as S1, S2, etc. The text is then translated into the formal representation of the discourse structure with (or without) the assistance of a natural language compiler and a linguistics consultant. The relationship between each sentence of the original text and the corresponding elements of the discourse structure is preserved by means of the sentence-tags. As the translation of the text continues, each word is entered in the lexicon. This lexicon is used to store dictionary definitions for the words used in the text and to store word-associated data required by the generator. Together, the lexicon and discourse structure form the ITl data base. The addition of a generative grammar to be used in sentence generation completes the preparation of an ITl data structure.

The tutorial operation of ITl is demonstrated in the protocol of Figure 2. The student has read the assigned text but has had some difficulty with it. Perhaps the syntax was unclear or there was doubt concerning the exact meaning of some word used in one of the sentences. If the student wants to review these troublesome points before undertaking a quiz, he consults the text and then types

(EXPLAIN S4)

```
STOP)))←←←
(ENTER COMMAND)


STUDENT
*STUDENT*MODE*
***
(EXPLAIN S1)
S1
(THE PLANTS ARE THE STATELY GRACEFUL COCONUT-PALM)
(THEY FIRST APPEAR ON A NEWLY FORMED TROPICAL ISLAND)
*END*EXPLAIN
***
(EXPLAIN S4)
S4
(THE SEED IS PECULIARLY ADAPTED FOR DISTRIBUTION)
(OCEAN WAVES AND OCEAN CURRENTS CARRY IT OVER A THOUSAND MILES
FROM THE PARENT PLANT)
(THEY DISTRIBUTE IT)
(IT GROWS ON SOME DISTANT SHORE)
*END*EXPLAIN
***
(DEFINE COMMERCE S3)
(IT1 5)
(WORD IS NOT USED IN SENTENCE)
***
(DEFINE COMMERCE S2)
COMMERCE
(N - THE SALE OR EXCHANGE OF GOODS - TRADE - BUSINESS)
***
(EXPLAIN S3)
S3
(THE SEED *S THICK FIBROUS HUSK AND ITS HARD SHELL SECURELY
PROTECT IT FROM THE ACTION OF SEAWATER)
*END*EXPLAIN
***
(EXPLAIN S2)
S2
(THE SEED IS THE COCONUT OF COMMERCE)
*END*EXPLAIN
***
```

FIGURE 2.--Sample Protocol

```
(QUIZ S1 S3)
(ANSWER TRUE OR FALSE)
(THE PLANTS ARE THE STATELY GRACEFUL COCONUT-PALM)
TRUE
*RIGHT*
(THEY FIRST APPEAR ON A NEWLY FORMED TROPICAL ISLAND)
TRUE
*RIGHT*
(THE SEED IS THE COCONUT OF COMMERCE)
TRUE
*RIGHT*
(ITS THICK FIBROUS HUSK AND ITS HARD CORE SECURELY PROTECT IT
FROM THE ACTION OF SEAWATER)
FALSE
*RIGHT*
(YOUR SCORE WAS 4)
(TOTAL POSSIBLE 4)
***
***
(QUIZ S5 S8)
(TYPE THE MISSING WORD)
(ADAPTATION ACCOUNTS FOR THE SEED *S WIDE ------)
DISTRIBUTION
*RIGHT*
(THE STATELY GRACEFUL COCONUT-PALM ORIGINALLY WAS NATIVE TO
CERTAIN ISLANDS OF THE ------ OR OF TROPICAL AMERICA)
PACIFIC
WRONG
(ANSWER IS INDIAN*OCEAN)
(IT IS NOW FOUND ON MOST TROPICAL SHORES IN THE ------ AND THE
WEST*INDIES)
**
(YOUR SCORE WAS 1)
(TOTAL POSSIBLE 2)
***
STOP
```

Note.--The notation ** terminates the quiz.

FIGURE 2 (continued).--Sample Protocol

to command the explanation of the sentence. The ITl responds by generating a set of simple sentences which preserve the meaning of the original sentence, S4.

After requesting other "explanations," the student finds that the meaning of "commerce" is unclear in the context of a sentence, so the student enters

(DEFINE COMMERCE S3)

to obtain a definition of the word as used. The system finds that the word is not used in S3, and displays an appropriate diagnostic response. The student then enters

(DEFINE COMMERCE S2)

which results in the printing of a dictionary definition for the sense of the word used in sentence 2.

The student browses through the text briefly and is then ready for a quiz. He enters

(QUIZ S1 S4)

to limit the scope of the quiz to the first four sentences in the text. On the basis of a random selection, ITl decides to use true-false questions in the quiz and so advises the student. The quiz is then printed, one question at a time, in the form of simple declarative sentences in each of which one noun may have been replaced by another noun which renders the sentence false. After each question has been presented to the student, ITl accepts a one-word answer; compares it with the correct response, and prints an appropriate message. The system accumulates the student's score as the quiz progresses. After the last question has been answered, ITl prints the score and then awaits further instructions.

The student enters

(QUIZ S5 S8)

to invoke a test over the indicated passage. This time the system elects to produce fill-in-the-blanks questions, in which one noun is replaced by a blank. As each response is entered, it is compared with the word which was omitted, and an appropriate response is generated. Tiring of the activity, the student enters

"**"

to end the quiz, then

STOP

to terminate the session.

This exchange demonstrates the three basic student commands implemented in this first ITl system. Some extensions are discussed later.

In addition to the *student mode*, ITl may operate in a *supervisor mode* for program testing and data base editing. In this mode the LISP interpreter is essentially placed at the supervisor's disposal, and he may invoke any LISP function. Several conversational utility routines, used in editing the data base, may be entered in this way. The supervisor may also change program variables or may even define new LISP functions in the supervisory mode.

THE TUTORIAL SUBSYSTEM: SUPERVISOR AND CONTROL FUNCTIONS

The modularity of IT1 is preserved in the tutorial *executive* itself. The executive consists of a simple *supervisor*, a *command-accepting function*, and a set of *command functions*. The basic function of the supervisor is outlined below.

Each student command consists of a *command word* and an *argument list* (e.g., QUIZ S1 S5). A command function and a dummy argument list are associated with each command word. The executive calls a command-accepting function, ACCEPT-CMD, to acquire the student command and execute preliminary validity tests. If the command is accepted, the associated command function name is retrieved, and the real arguments are substituted for the dummy arguments. The command function is then executed. It supervises its own student communications, tests for valid arguments, and returns control to the executive when finished. The present command list is as follows:

Command	*Command Function*
(EXPLAIN Si)	EXPLAIN*
(QUIZ Si Sj)	QUIZ*
(DEFINE word Si)	DEFINE*
STOP	STOP

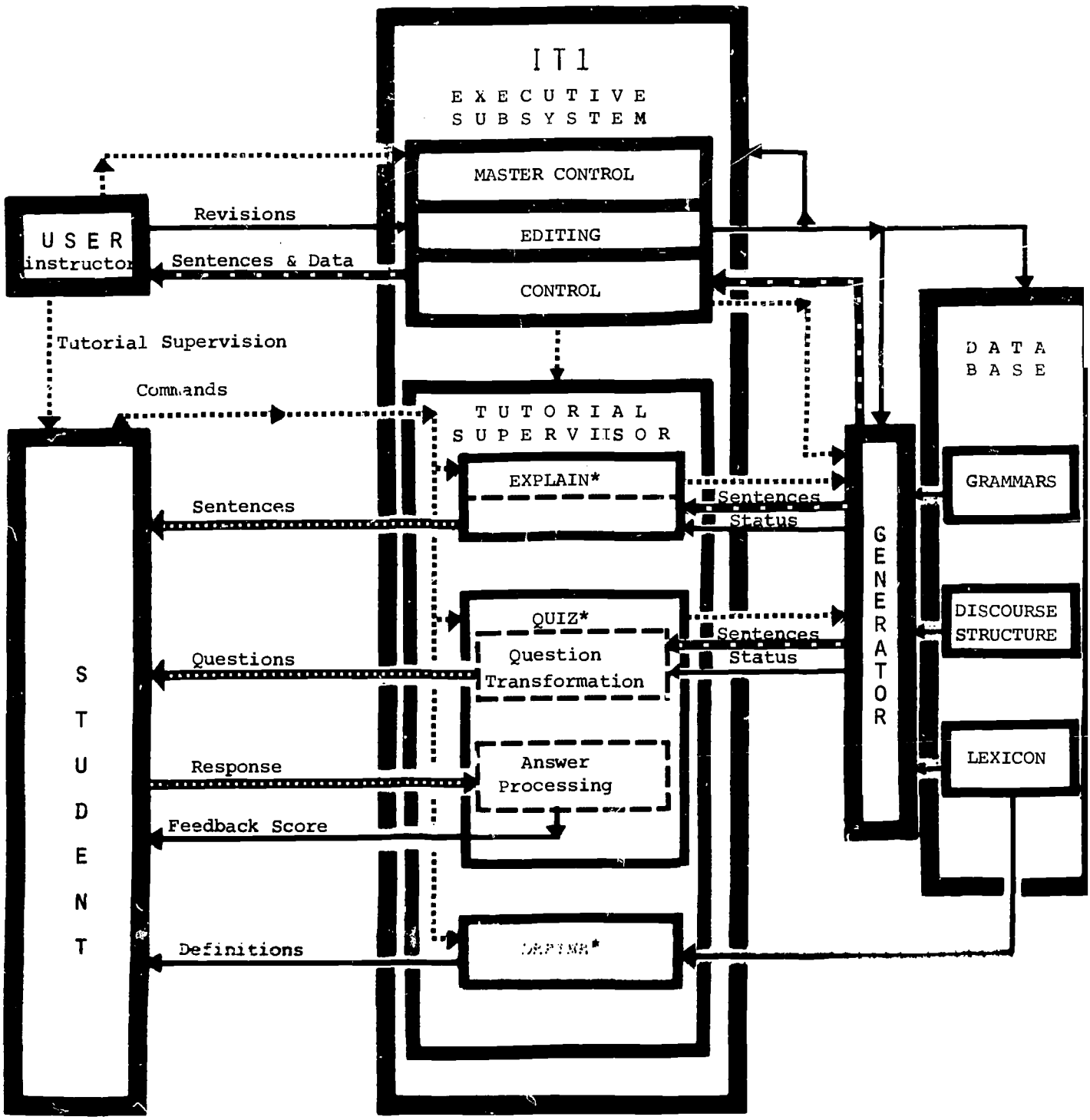
The command function STOP is a special case used to terminate a session. Eventually, a STOP function will be added which will create and file a summary of the student's performance during the session, but this feature has not been implemented in IT1.

The interactions among student, instructor, and the various components of ITl are summarized in Figure 3. The control functions are discussed below. Certain details of their interaction with the generator and data base will necessarily remain unclear until those components have been described in more detail.

In the following discussion, the term *user* refers to a course designer, programmer, or other persons who may modify the program or its data structure. The term *student* has the usual meaning of one who is being taught and, in particular, one whose interaction with the system is limited to the student commands listed above.

The Command EXPLAIN Si. EXPLAIN* takes as an argument the alleged sentence-tag specified by the student in the original command. If the argument is not in fact a sentence-tag, EXPLAIN* prints a diagnostic message and returns. Otherwise, EXPLAIN* calls the generator to produce the appropriate set of sentences. At the time that the text was compiled, the sentence-tag was used to preserve the relationship between the original sentence and the corresponding objects in the discourse structure. This was done by binding the attribute HDVB to each sentence-tag. The attribute HDVB indicates the object in the discourse structure which was created from the head verb in the original sentence. This binding is retrieved by EXPLAIN* which then calls the generator to produce a sentence from the indicated object. The sentence is returned to EXPLAIN* as the value of the generative function. The example in Figure 2 (see pg. 12) shows that the explanation of a text sentence may require the generation of several simple sentences. To avoid confusion, the term *generated passage*, or simply *passage*, will be applied to the set of sentences generated by a single EXPLAIN* command. The

FIGURE 3. -- Generalized Block Diagram of ITl.



initial sentence of the generated passage is printed by EXPLAIN* which then determines (by means discussed later) whether the last possible sentence has been generated. If not, the generator is called to produce the next sentence of the passage, and EXPLAIN* prints it. This process continues until the passage is complete.

The Command DEFINE. DEFINE* is given two arguments corresponding to the word and sentence-tag specified by the student in the original command. A dictionary article for the word is retrieved from the lexicon and printed.

The Command QUIZ Si Sj. QUIZ* determines whether both sentence-tags are legitimate. If they are not, a diagnostic is printed. If both are legitimate, QUIZ* generates and scores a quiz corresponding to that segment of the text bounded by Si and Sj (Si = Sj is permitted, and Sj may precede Si in the text). Each question is a transformation of an ordinary generated sentence. Basically, QUIZ* produces a passage which is the union of all the passages which EXPLAIN* would produce for each of the text sentences Si, Si + 1, . . . , Sj. Answers are accepted and processed after each question is printed.

Questions are produced by simple substitutions in the original generated sentence. The basic procedure is to generate a sentence in the usual way and, then, to substitute for some word in the sentence a word which renders the sentence false, or a "blank" (i.e., _____) to produce true-false or fill-in-the-blank questions, respectively. A set of true-false or blanked questions is randomly selected; the user may control the frequency of choice. All questions in each quiz are of the same type.

The user may specify *a priori* the syntactic class of words which are to be candidates for substitution. Nouns were selected in the example of Figure 2. Any noun in the sentence may be replaced. Only one word is substituted in each sentence, and a particular word for substitution is selected at random. Thus, if there are four nouns in a sentence, four different blank questions may be derived from it. Falsification is randomly controlled, with a probability of 0.5 that falsification will occur in any given question. Thus, if there are two possible falsification substituents for each of the four nouns, the sentence may be transformed into one of eight possible false statements, and nine different true-false questions may be derived from the base sentence.

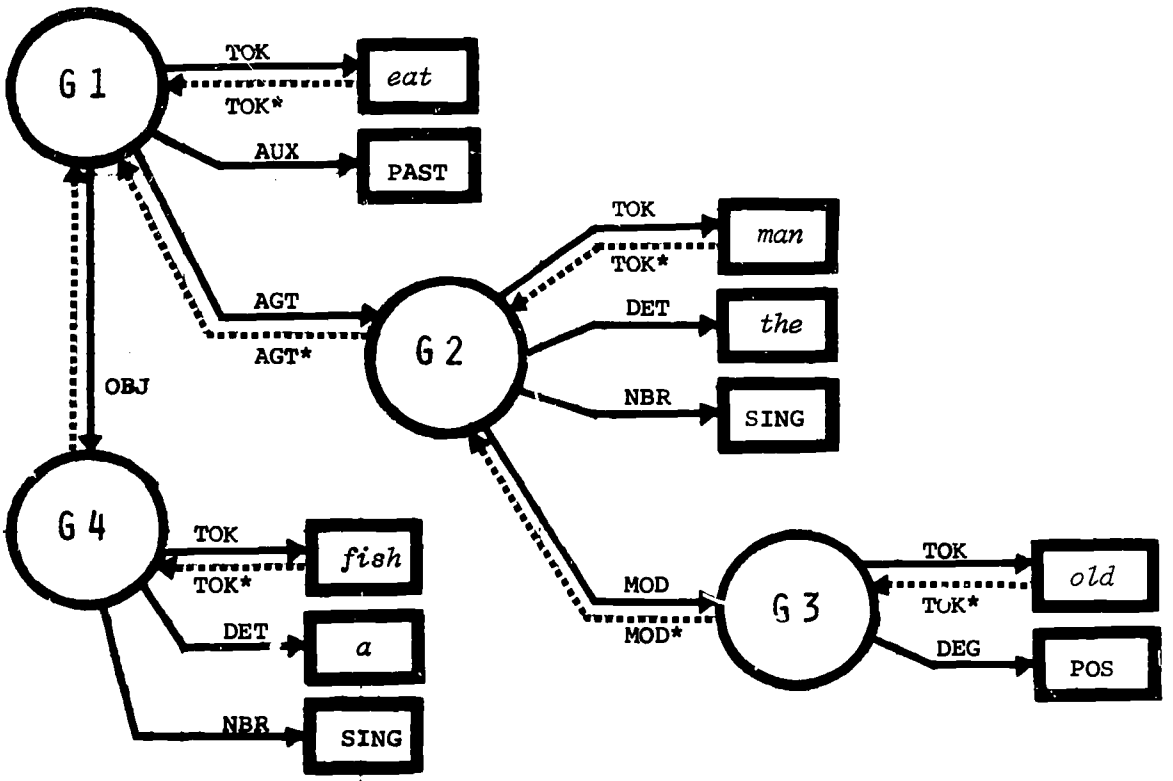
As each substitution is made, the correct answer is stored. In blank questions, the original word is stored; in the true-false questions, the truth value *true* or *false* is stored. The transformed sentence is printed, and student response is accepted. If the response is a double asterisk (**), the quiz is terminated. Otherwise, the response is compared with the stored answer, and an appropriate message is displayed to the student. The total of incorrect answers is accumulated, and after the last question, the score is printed.

THE DATA BASE:
DISCOURSE STRUCTURE AND LEXICON

Three kinds of data have been mentioned which must be provided in each ITI system: the *discourse structure*, the *lexicon*, and a *grammar*. The discourse structure and lexicon form one continuous structure, which will be referred to as the *data base*. The grammar is a separate structure containing rules which govern the derivation of sentences from the data base. In the following section, the grammar and generative algorithm are discussed in more detail, and the nature of their correspondence becomes clear. The general structure of the data base and the content chosen in this experimental implementation are described presently.

The structure of the data base is based on Quillian's (1966) notion of a *semantic network*, wherein each cognitive entity in a complex unit of meaning is defined by relations which exist between it and other cognitive entities. Such structures may be represented by a graph such as that of Figure 4a. Each entity is represented by a labeled node, and relations by labeled arcs. Simmons (1970a) has shown that the attribute-value lists of Figure 4b are equivalent to the graphical representation. They provide a convenient machine-readable representation for computational applications.

The node G1 represents an event, namely the eating of a fish by an old man. The fact that the event in question was a specific *token* of the larger class of events identified by the verb *eat* is specified by the TOK relation between G1 and the lexical entry *eat*. The AGT arc indicates the active participant in the event, G2, the man, and OBJ



A . — Semantic Network

(G1 (TOK *eat*) (AGT G2) (OBJ G4) (AUX PAST))

(G2 (TOK *man*) (MOD G3) (AGT* G1) (DET *the*) (NBR SING))

(G3 (TOK *old*) (MOD* G2) (DEG POS))

(G4 (TOK *fish*) (OBJ* G1) (DET *a*) (NBR SING))

B . -- Equivalent List Representation

S1 . *The old man ate a fish.*

C . — Text

FIGURE 4 . -- A Simple Discourse Network

indicates the participant receiving the action. The inverse relations TOK*, AGT*, etc., are represented in Figure 4a by a broken line. Lexical entries are represented in lower case letters, and node and arc labels are in upper case letters to distinguish them from lexical entries.

The arcs NBR and AUX are not true relations in the sense of the arcs mentioned above, since they do not indicate objects in the discourse structure or lexicon. The NBR arc indicates the syntactic number of an object, and AUX indicates information required to produce the appropriate inflection *ate* of the verb *eat*. These and similar relations are termed *features*. Relations and features are collectively termed *attributes*.

Each lexical entry is the head of a similar structure which defines that entry. The lexical substructures are omitted for the present.

The following conventions are used in reference to discourse networks:

(1) The node G1 is said to *dominate* G2, G4, and G3, since G1 is superior to the others in the hierarchy.

(2) G2 is the value of the attribute AGT in G1; therefore, G2 is said to be dominated by the relation AGT in G1 or, simply, AGT dominates G2 if the identity of the dominant node is of no interest.

(3) A node is said to be *headed* by the lexical entry indicated by the TOK of the node. Thus, G1 is headed by *eat*. Similarly, G1 is said to be *verb-headed* since its head is a verb.

The structure of Figures 4a and 4b was derived from the sentence in 4c, and might have been derived from any of the examples 1-6 of page 7 in the preceding section. It could have been derived mechanically from any of sentences 1-5, and this is the basis of the discourse structure used in IT1. Semantic networks are used as canonical forms from which a variety of English representations might be generated in a systematic way, so that a human derives approximately the same meaning from any of the representations. The operational notion of "meaning" is defined in terms of lexical content and syntactic form. If a given English string X is well formed and meaningful to a competent reader, then it defines a semantic equivalence class of strings. A different string Y is in the class of X if:

(1) All the content words in both X and Y are morphological derivatives of common lexical roots.

(2) The syntactic construction of Y preserves the meaning of X.

In the absence of any effective definition of "meaning" it is necessary to illustrate the intent of the syntactic restriction by an example. Consider the following strings, all of which might be derived from any one of the set:

(1) The man ate the fish.

(2) The fish was eaten by the man.

(3) . . . the eating of the fish by the man . . .

(4) The fish ate the man.

Strings 1-3 are in the same semantic class. String 4 is not in that class, even though its lexical content is identical with that of the others. The syntactic form of 4 leads to quite a different interpretation.

In 3, the "man-eat-fish" idea is subsumed by a larger sentence, while it is given independent syntactic status as a sentence in 1 and 2. Thus semantic class membership is partially independent of surface syntactic classification.

The generative process takes place in a two-dimensional environment in which the lexical content of a derived string must be preserved while the commonly accepted forms of English are imposed upon the string in such a way that the derived string remains in the semantic class of its base in the original text. The operational function of the data base is to provide sufficient information to permit the definition of effective procedures for generation of sentences which conform to these restrictions.

The structure presented here is based on that used by Simmons (1970). Part of its content was suggested by the work of Fillmore (1969) concerning the content of the lexical component of linguistic systems.

The attributes used in the IT1 discourse structure are defined in Table 1. The remarks indicate the application of each attribute as well as the types of syntactic phrases which are derived from them. The derivations are discussed in a later section. Examples of the data base used in the first implementation of IT1 are given in Appendix A.

In the present system, conjoined nodes are represented by a node which is headed by a conjunction which dominates the conjoined elements by the relations A1 and A2. An example (X2) is presented in Figure 5.

Prepositions may be represented in two ways. In general, locative, translational, and similar prepositions are represented explicitly in

TABLE 1

DISCOURSE ATTRIBUTES

Name	*Value Type*	*Remarks*
TOK	Lexical Entry	See page 21.
AGT, OBJ DAT, INSTR	Discourse Node	Indicate nominal objects which have, respectively, the agentive, objective, or instrumental relations to a verb-headed node.
NOM PNOM PADJ	Discourse Node	Indicate nominative, noun-complement, or adjective-complement of *to be*.
TIME MANN LOC	Discourse Node	Modifiers of verb-headed nodes, specifying time, manner, and location, respectively. LOC and TIME usually dominate a preposition-headed node; MANN may dominate adverb- or preposition-headed nodes.
MOD	Discourse Node	Adjectual or adverbial modifiers which are not represented by any explicit relational attribute.
PMOD	Discourse Node	Indicates a relation normally expressed by the preposition which heads the node dominated by PMOD.
POBJ	Discourse Node	Dominates the object of a preposition-headed node.
A1, A2	Discourse Node	Indicates co-nodes of conjoined nodes.
DET	Lexical Entry	Indicates a determining article, etc.
NBR	Number Indicator	S = Singular; PL = Plural.
AUX	Tense Indicator	PR = Present; PAST = past; compound tenses are constructed syntactically.
DEG	Degree Indicator for Adjective	POS = Positive; COMP = Comparative; SUPR = Superlative.

Jack and Jill ran up the hill.

(X1 (TOK *run*) (AGT X2) (LOC X5) (AUX PAST))

(X2 (TOK *and*) (NBR PL) (A1 X3) (A2 X4) (AGT* X1))

(X3 (TOK *Jack*) (A1* X2))

(X4 (TOK *Jill*) (A2* X2))

(X5 (TOK *up*) (POBJ X6) (LOC* X1))

(X6 (TOK *hill*) (DET *the*) (NBR S) (POBJ* X5))

FIGURE 5. — Conjoined and Prepositional Discourse Nodes.

the discourse network by a node such as X5 (see Figure 5), which is dominated by LOC and headed by a preposition. Other such nodes are dominated by PMOD. On the other hand, the prepositions *by*, *of*, and others which are optionally used to represent relations between verbal and nominal objects, are not represented explicitly. Each such relation label is given the attribute RELPREP, which indicates the print image of the appropriate preposition. In generation, a function retrieves the preposition when it is required in some syntactic environment. In practice the alternate representations are equivalent in the sense that a prepositional phrase or a simple noun phrase may be derived from either.

The Lexicon

The lexicon supports both generative and tutorial operations. Lexical data are used to supply syntactic information and the actual print-images for words to be used in generated sentences. The lexicon also includes definitions for use by the DEFINE* function and falsification substitutes for use in production of True-False questions.

The lexicon consists of a collection of lexical entries. An entry is composed of a head and its associated attributes and their values. Each entry corresponds to one sense of an English word. For example, the word *plant* might be used to mean any one of the following:

plant 0: n; a living organism of the vegetable kingdom.

plant 1: n; a factory or mill.

plant 2: vt; to bury a seed or cutting in the ground for growth.

Such a collection of entries, all of which are represented by the same word, is represented in the lexicon in a manner suggested by the

example. One entry is headed by the word itself (the *0th* entry). Each of the remaining entries is headed by a symbol coined by concatenating a digit with the word, e.g., *plant*1. It is this *subscripted* symbol which is indicated by the TOK relation in the discourse structure and which is used by the generator as an entry into the lexicon for terminal operations. Each subscripted word is linked to the parent unsubscripted word by the attribute SENSES and its inverse, SENSES*, as indicated below:

(plant (SENSES (plant plant1 plant2)) (SENSES* plant) . . .)

(plant1 (SENSES* plant) . . .)

(plant2 (SENSES* plant) . . .)

The SENSES* attribute is used to fetch the unsubscripted form as a print image for all entries. Each entry contains the set of attributes and values required for the computational applications. Since IT1 lacks semantic operations such as question-answering and analytic capacity, the linguistic function of the IT1 lexicon is restricted to the storage of syntactic features and morphological data in the form of *print-images* which correspond to the various inflections of the root word which heads each entry. These inflected forms are used, for example, to indicate the number of a noun or to obey the rule of number agreement between a verb and its surface subject.

The usual parts attributed to each verb are: infinitive, present, past, past participle, and present participle. The third-person singular is also explicitly specified. Verbs which are irregular in the formation of the plural third-person past-tense have that form explicitly specified also and are marked with the flag IREG. Some verbs, such as *appear* and *adapt*, are closely related to nouns (*appearance* and *adaptation*) which are used in place of the usual gerund in some circumstances. These

nouns are also associated with the verb. The attribute-names used to indicate these print images are tabulated in Table 2. The head itself is the infinitive, and hence, no explicit specification of this form is required.

Noun entries are headed by the singular form and the plural form is indicated by the attribute PLF. The genitive is synthesized by the generator from the singular by adding the 's ending. Adjective entries are headed by the positive form, with comparative and superlative indicated by COMP and SUPR, respectively. Adverb entries are provided independently of adjectival roots. Prepositions, conjunctions, articles, and pronouns are either represented directly in the discourse structure or are synthesized in the generation process and are not included in the initial lexicon. Several additional attributes are used by the tutorial subsystem. These attributes and the application of each are given in Table 3. Except when specified otherwise, these attributes are used in all entries.

The lexicon is stored in a network structure similar to the discourse structure. Sample entries are given in Figure 6 in the list notation.

TABLE 2

LEXICAL ATTRIBUTES USED IN GENERATION

Attribute Name	*Value (Print-Image for)*
PRSF	Third person singular, present tense
PAST	Regular past tense
PRPART	Present participle
PPART	Past participle
NOUN	Derived noun

TABLE 3

LEXICAL ATTRIBUTES USED IN TUTORIAL SYSTEM FUNCTIONS

Attribute	*Value*
LEX	A dictionary article for the given word-sense printed by the executive in response to the DEFINE command.
USED-IN	A list of the tags of all sentences in which the morpheme is used. Used by the executive to determine the appropriate definition in responding to the DEFINE command.
TOK*	A list of discourse nodes in which the head is the value of TOK.
FALS	A list of words which may be substituted for the head in sentences in order to render the sentence false. Used only in noun entries in IT1.

```
(COCONUT-PALM ((SENSES COCONUT-PALM) (FALS OAK ELM DATE-PALM FIG)
(USED-IN S6) (PLF . COCONUT-PALMS) (LEX N; A TALL SLENDER
TROPICAL PALM COCOS NUCIFERA) (TOK* . G3) ( SENSES* .
COCONUT-PALM)))
(APPEAR ((SENSES APPEAR) (LEX VI; TO BECOME VISIBLE) (FALS LEAVE)
(PRSF . APPEARS) (PASTF . APPEARED) (PPART . APPEARED) (PRPART.
APPEARING) (PRPART . APPEARING) (TOK* . G10) (SENSES* . APPEAR)))
(BE ((SENSES BE BE1) (USED-IN S1 S2 S3) (IREG . T) (LEX V ; THE
SAME AS ONE OF) (PRSF . IS) (PRPART . BEING) (PRPLF . ARE)
(PSPLF . WERE) (PPART . BEEN) (PASF . WAS) (TOK* G100 G70 G1 G20)
(SENSES* . BE)))
(BE1 ((USED-IN S2 S3 S4) (LEX V; TO HAVE THE PROPERTY OF (HIS
EYES ARE BLUE) IN A STATE OF (IT IS ADAPTED)) (SENSES* . BE)))
(FIND ((SENSES FIND) (USED-IN S6) (FALS LOSE) (LEX VT ; TO COME
UPON BY CHANCE; TO MEET WITH) (PRSF . FINDS) (PASTF . FOUND)
(PRPART . FINDING) (TOK* . G81) (SENSES* . FIND)))
(STATELY ((COMPR . STATLIER) (SUPR . STATLIEST) (SENSES STATELY)
(LEX ADJ) (TOK* . G7) (SENSES* . STATELY)))
(ISLAND ((SENSES ISLAND) (FALS MOUNTAIN RIVER) (USED-IN S1 S6 S7
S8) (PLF . ISLANDS) (LEX N - A TRACT OF LAND COMPLETELY
SURROUNDED BY WATER BUT TOO SMALL TO BE CALLED A CONTINENT)
(TOK* G110 G97 G12 G73) (SENSES* . ISLAND)))
```

FIGURE 6.--Lexical Entries

THE GENERATIVE SYSTEM

The Syntax Directed Generator

The tutorial function of the generative system is the derivation of sets of sentences corresponding to some text segment specified by an EXPLAIN* or QUIZ* command. These derived sentences must belong to the semantic class of the specified text segment. Since the original phrase structure of the text is lost in the discourse network, there is no simple correspondence between the discourse nodes and text segments. At the same time the lexical content of every text segment is reflected in some subnet of the discourse network, so that the lexical content of the text is accessible to retrieval by well-defined processes. The generative strategy employed in ITl is to define generalized derivational procedures which produce phrases that preserve the meaning of the original text syntactically, while guaranteeing the retrieval of at least the full lexical content of the original text. Other processes are imposed on this basic generative process to restrict the lexical content of a generated passage to just that of some specified text segment. The control processes will be discussed after the basic generative system is described.

The generative system is defined in terms of the derivation of a *phrase* from some *root* node in the discourse network. A phrase is recursively defined to be either

(1) a word

(2) a sequence of phrases

In the latter case, the defined phrase is said to be the *parent* phrase and the individual members of the defining sequence are referred to as

dependent phrases. Similarly, the root of the parent phrase is the parent root, and the roots of dependent phrases are dependent roots. A dependent phrase may be derived from the parent root; i.e., every phrase (root) is its own parent and its own dependent.

The system is syntax-directed in that the derivational process is defined by a *generative grammar* supplied by the user, which is interpreted by a generative algorithm. Syntactic construction is described in the usual manner of phrase-structure grammars. The lexical content of the root is preserved by dynamic definition of dependent roots in terms of the attributes of the parent root. The grammar provides for definition of conditions upon which the selection of syntactic productions is predicated. For a simple subset of English it suffices to define these conditions in terms of the attributes of the parent root. The sentence-rule below illustrates these properties of the basic rule format.

```
S → [AGT OBJ] (SUBJ AGT)(VP *)(NP OBJ)/
    [AGT] (SUBJ AGT)(VP *)/
    [OBJ] (SUBJ OBJ)(PASVP *)
```

The rule defines three alternate productions for a sentence. Each production is predicated on a condition which is represented by the square-bracketed sequence of attribute-names. A condition is true if it is empty (i.e., [ ]) or if the root has every attribute named in the condition. Thus, if the sentence-root has both AGT and OBJ attributes, the first production is applied; otherwise, if the root has the AGT attribute, the second production is applied, and so on. If none of the conditions are true the "null" production is applied and results in generation of the empty string containing no words.

Each production is a sequence of *pairs* (x,y), where x is either an attribute-name or a nonterminal symbol which defines the syntactic class of a dependent phrase, and y defines the dependent root. If y is the symbol "*", then the dependent phrase is derived from the parent root; otherwise y must be an attribute-name and the dependent phrase is derived from the discourse node which is dominated by the attribute y in the parent root.

If x is a nonterminal, then the pair is said to be *nonterminal*. Otherwise, the pair is said to be *terminal*, and its *generative value* is the value of the attribute x in the dependent root. For example, the article and uninflected noun are produced by terminal pairs in a simple class of noun phrases defined by the rule

NP → [ ] (DET *)(TOK *).

Terminal pairs correspond to the initial step in the recursive definition of a phrase, and nonterminal pairs correspond to the recursive step. To complete the analogy, the generative value of a nonterminal pair is the result of concatenating the generative values of dependent phrases derived from it. That is, if a pair (NP *) were applied to G2 in the discourse network of Figure 5, the result would be computed in the following steps:

(NP *) → ((DET *)(TOK *)) → (the (TOK *)) → (the man).

This simple NP-rule does not preserve the lexical content of its root, since it provides no means for expressing adjectival modifiers or number, to name just two deficiencies.

Some power to express transformational operation, synthesis of function-words, and more complex conditions is necessary for pronominalization, subject-verb agreement, and other syntactic properties of the desired subset of English. This power is provided by adding functions to the generative system, permitting function-names to appear in grammar rules, and attaching functions to certain nonterminal symbols. These functions are classified according to the use which is made of them.

(1) *Pseudo-functions* are attached to nonterminal symbols. They are used only for their transformational effect on the discourse network, to set global registers, and the like. They have no generative value, and their functional value is ignored by the generative algorithm. A pseudo-function is attached to the nonterminal SUBJ to retrieve the attribute NBR and its value from the SUBJ-root, and insert them into the root node of the parent sentence for later use in enforcing subject-verb agreement when the verb-phrase is derived from the nonterminal VP.

(2) *Predicates* are used to augment the basic conditions described above. An example is PLP which is true if the root has the attribute-value pair (NBR PL), i.e., if the root is plural, and otherwise false. Predicates may also have useful side effects as in the case of MARKF which is described later.

(3) *Terminal functions* are used for synthesis of function words, punctuation, and pronouns, and may also be used for effects similar to those described in (1) above.

The application of functions in the grammar and the interpretation of rules is defined in terms of a generative algorithm. This algorithm and its auxiliary functions are described in an informal language which borrows features from both the ALGOL and LISP meta-languages, with some simple conditions written out in English.

A grammar G is a collection of rules of the form

$$T \rightarrow C_1P_1/C_2P_2/ \ldots /C_nP_n$$

where T is a nonterminal symbol. Each C_i is a condition of the form

$$[c_{1i}\ c_{2i} \ldots c_{m_i}]\ ;\ m_i \geq 0$$

where each c_{ji} is either an attribute-name or the name of a predicate.

Each production P_i is a sequence of pairs

$$((x_{1i}\ y_{2i})(x_{1i}\ y_{2i}) \ldots (x_{k_i}\ y_{k_i}));\ k_i \geq 0$$

where each x_{ji} is either an attribute-name, a nonterminal symbol, or the name of a terminal function, and each y_{ji} is either an attribute-name or the symbol "*".

The set of rules which compromises a grammar G is defined in the following manner. Each nonterminal symbol t has the attribute G, and its value is the sequence of condition-production pairs which defines t in G. The function *getr* is used in the generative process to retrieve the generative definition of a nonterminal. That is,

$$getr[t;G] = IF\ t \text{ has the attribute G } THEN\ (C_1P_1/ \ldots /C_nP_n)\ ELSE\ nil$$

The recursive generation process is initially applied to some specified root node and nonterminal symbol. The appropriate production for the given root and nonterminal is selected, the successive pairs of the

production are evaluated, and the results concatenated to produce the desired string. Production-selection for an arbitrary discourse node r and nonterminal symbol t is carried out by the functions *select*, which is applied to the sequence of condition-production pairs returned by *getr* and *gcond* which evaluates conditions.

gcond [r; (c_1 c_2 . . . c_m)] =
  IF m = 0 *THEN* true
    ELSE IF c_1 is an attribute name
      THEN IF r has the attribute c *THEN* *gcond* [r; (c_2 . . . c_m)]
        ELSE false
      ELSE IF c_1[r] *THEN* *gcond* [r; (c_2 . . . c_m)]
        ELSE false

select [r; (C_1P_1/C_2P_2 . . . $/C_nP_n$] =
  IF n = 0 *THEN* nil
    ELSE IF *gcond* [r; C_1] *THEN* P_1
      ELSE *select* [r; ($C_2P_2/$ . . . $/C_nP_n$)]

The utility function *gassoc* is used to return the value of an attribute a in a discourse node r. For the purpose of this discussion a discourse node is identified with its defining attribute-value list,

$((a_1v_1)(a_2v_2)$ ... $(a_nv_n))$

The latter, more explicit notation will be used when it can contribute to clarity.

gassoc [a; $((a_1v_1)(a_2v_2) \ldots (a_nv_n))$] =

IF n = 0 *THEN* nil

ELSE IF a = a_1 *THEN* v_1

ELSE gassoc [a; $((a_2v_2))$]

Generative evaluation is carried out by the following set of recursive functions which evaluate successive sub-elements of productions. The process is initiated by application of *gen1* to an initial root node *r* and nonterminal symbol *t*, where the derivation is governed by grammar G.

gen1 [r;t] = *gen2* [r; *select* [r; *getr* [t; G]]]

Next, *gen2* transmits successive pairs of the production returned by select to *gen3* for evaluation, and concatenates the resulting dependent phrases.

gen2 [r; $((x_1y_1)(x_2y_2) \ldots (x_ny_n))$] =

IF n = 0 *THEN* nil

ELSE conc [*gen3* [r; (x_1y_1)]; *gen2* [r; $((x_2y_2) \ldots (x_ny_n))$]]

Dependent roots are selected by *gen3* which applies *teval* to evaluate terminal pairs, or re-initiates the process by applying genl to nonterminal pairs.

gen3 [r; (x y)] =

IF getr [r; G] *THEN*

1. x[r;y];
2. *gen3*:= *IF* y = "*" *THEN gen1* [r; x]

ELSE gen1 [*gassoc* [y; r]; x]

ELSE IF y = "*" *THEN teval* [x; y; r]

ELSE teval [x; y; *gassoc* [y; r]]

Note that the nonterminal recursive step 2 above is preceded by application of the pseudo-function attached to x, to the current root r in step 1, but the value, if any, of the pseudo-function is not appended to the surface string. If no pseudo-function is attached to x, step 1 has no effect.

Finally, *teval* applies x to the root r and relation y if x is a terminal function or retrieves its value from the root node if it is an attribute-name.

teval [x; y; r] = *IF* x is a terminal function *THEN* x[y; r]
ELSE gassoc [x; r]

Pronominalization

A brief reference was made earlier to higher-order principles which govern the derivation of multi-sentence passages from large data bases. Similar processes govern the natural use of pronouns. We are so habituated to the use of pronouns that passages generated without them seem almost hopelessly clumsy. Since the minimal goals of the IT1 system might be compromised by omission of this important detail, a simple mechanism was implemented which seems to satisfy the minimum requirements.

The present pronominalization mechanism permits the substitution of pronouns in the third person for modified noun phrases under restricted conditions based on the content of the current passage. If substitution is not permitted, the full noun phrase is produced with all possible modifiers.

This characteristic leads to tiresome repetition in long passages, but it is not especially annoying in the ITl application since the control functions limit the length of passages to a few sentences. Pronoun substitution makes three requirements of the generative system:

(1) The eligibility of a discourse node for pronominalization must be determined.

(2) The surface syntactic case in which the pronoun appears must be specified.

(3) A pronoun must be produced which agrees with its antecedent in number and gender and which is inflected according to case.

These operations are carried out in the following way: Provisional eligibility is established by a pseudo-function bound to the nonterminal symbol NOUN, which is invariably applied to the root of a noun-phrase derivation. This pseudo-function adds the attribute PRON to the root. All noun-phrase rules test for the presence of this attribute. If it is absent, the full noun phrase is produced; if it is present, a function is executed to determine the eligibility of the current root for pronominalization and to attach the appropriate pronoun to the root if the substitution is permitted. Substitution is permitted only if the selected pronoun has not already been bound to a different antecedent. If substitution is blocked, the PRON flag is removed from the current node, and the original antecedent binding is removed from the pronoun which would otherwise have been used in substitution. The pronoun is thereafter free for substitution for any suitable noun phrase.

A minor complexity is introduced by the fact that antecedents are not bound directly to surface pronouns but to a root-form of the pronoun instead, which is inflected to indicate syntactic case. It is also necessary

to introduce surface case syntactically during derivation, since the discourse relations do not necessarily indicate the surface case of generated noun phrases. Case is introduced by means of three noun-phrase rules: NNP, ONP, and GNP, corresponding to nominative, objective, and genitive cases, respectively. Surface subjects are derived from the nonterminal SUBJ and NNP's; direct and indirect (e.g., dative) objects are generated as ONP's, as are prepositional objects. The GNP rule is used to produce possessive pronouns and adjectives. The interrelations of case, syntactic features, roots, and surface forms of pronouns are given in Figure 7. This table is represented internally by a network structure. Antecedents are bound to the ROOT by the attribute ANT. The pronoun-generating functions operate on this network to determine the root of the prospective pronoun and to find the current antecedent bound thereto. Each of the three case-preserving noun-phrase rules calls a separate pronoun-generating function. Following the naming convention of the noun-phrase rules, these functions are termed NPRON, OPRON, and GPRON. Each of these functions preserves the case induced by the grammar rule which invokes it.

Figure 8 is an example of the application of these functions in production of nominative noun phrases or pronouns and, incidentally, an example of the method of generating conjoined constituents. The NNP rule tests for the PRON mark; if the mark is present, NPRON is first applied and then the NNP1-rule is applied. The NNP1-rule tests for the PRON flag again. If the pronoun substitution was permitted by NPRON, the appropriate pronoun is attached to the root of the NNP expansion by the attribute PRON, and becomes the generative value of NNP1. If NPRON blocked the pronoun substitution, the PRON attribute is removed, the PRON condition on the NNP1-rule

Syntactic Features			Case			
Root	Gender	Number	Nominative	Objective	Genitive	Possessive Adjective
HE*	M	S	he	him	his	his
SH*	F	S	she	her	hers	her
IT*	N	S	it	it	---	its
TH*	N	PL	they	them	theirs	their

FIGURE 7 . — Inflection of Pronouns

```
NNP  → [PRON] (NPRON *) (NNP1 *)/
       [A1] (DET *) (MODP *) (NNP A1) (TOK *) (NNP A2) (PP PMOD)/
       [AUX] (NPS *)/[] (NNP *)
NNP1 → [PRON] (PRON *)/[] (NNP *)
```

FIGURE 8 . —Nominative Noun Phrase Derivation.

will fail, and the NNP-rule is applied again unconditionally. On this second application of NNP, the [PRON] condition will fail, and the other alternatives will be applied. The [A1] condition is used to determine whether the root is a conjoined constituent. Note that the given rule order permits conjoined constituents to be pronominalized jointly, or each individual co-constituent may be pronominalized independently. The [AUX] condition in the third right-hand side is used to determine whether the root is verb-headed. If it is, a special noun-phrase rule schema headed by NPS is applied to produce a nominalized sentence. Finally, if all other conditions have failed, an uninflected noun phrase is produced by application of the NP rule. The rule schemata for the other syntactic cases are similar, except that in the genitive case, the noun is inflected by addition of 's in the full noun phrase.

Determination of Content

The preceding sections have shown how the generative subsystem controls content and form of strings of and below the order of simple sentences. The basic definition of the EXPLAIN command implies that several simple sentences might be required to contain the content of one complex sentence of the original text. Thus, the system must control content of passages, which are strings of higher order than simple sentences. The performance required for the EXPLAIN command is a restricted special case of the performance required of the generator as a separate entity. That is, the generator must produce (in response to one initial request to derive a sentence from a given node in the discourse network) every possible sentence that can be derived from the structure. This property is predicated on the

requirement that the discourse network be well-formed in the sense that every node is related to every other node through some sequence of node-relation-node paths.

The basic requirement in determining content is that some syntactic object is to be derived from every discourse node that is related in any way to the root of the initial sentence. Each time a string is derived, its root is examined to determine whether it is related to any unsatisfied node, i.e., if the root is directly related to any other node which was not the root of a substring of the current string. For every alternate production for expansion of a nonterminal, the set of attributes which dominate such nodes is, at worst, the complement of the set of attributes which dominate the roots of substrings in the expansion in question, with respect to the universe of all possible discourse relations in a given data base. A *supplementary* sentence is derived from every unsatisfied node. Then, if this procedure is applied at each step in derivation and if the grammar is general in the sense that a simple sentence can be derived from any node in the network, the generator will have the desired property. It will exhibit the desired behavior if no sentence is repeated. The MARKF predicate is used to assure this condition to avoid infinite repetition of a subset of the possible sentences.

Additional restrictions exist. Clearly, derivation of the supplementary sentences cannot be initiated until the current sentence is completed. Construction of a grammar which has this property and which exhaustively derives sentences from all possible supplementary nodes is tedious, at best, and has been done only for elementary data bases. The task was simplified for the present application by taking advantage of the character of

the data base and of the simple-sentence grammar and by introducing two functions to identify potential supplementary sentence-roots and to generate sentences from them at the appropriate time. The simple-sentence grammar was constructed to derive sentences from verb-headed nodes. In particular, it was constructed so as to derive a substring from every node dominated by a sentence-root through any chain of forward relations (e.g., AGT, as opposed to the inverse AGT*). This task is comparatively simple, since there is an inherent division of discourse nodes into verbal, nominal, and other classes, each of which is restricted in the types of attributes possessed by a node belonging to the class. This grammar was augmented by two additional functions, PUSH and POP, which are used in the following way:

> PUSH is applied in each noun-phrase derivation. By means of the inverses of verb-noun relations (AGT*, OBJ*, etc.), PUSH collects a list of the nodes which dominated the root of the noun-phrase by one of the verb-noun relation. A list of such nodes is then appended to a global list which is, in effect, a queue of prospective supplementary sentence-roots.
>
> POP is called when the current sentence is complete. It retrieves the queue and calls the generator to derive a sentence from each node in the queue.

Since PUSH is applied to every noun-phrase root and POP is applied to every sentence, additional supplementary roots are queued and used throughout the process. This process does not guarantee exhaustion of every well-formed data base, since only verb-noun relations are considered in the queueing process, but it suffices for the purposes of ITl.

Interaction with Control Functions

The preceding section essentially describes the communications between the control functions and the generator. In IT1, the POP function is omitted, and its task is performed by EXPLAIN* and QUIZ*. Both of the control functions carry out essentially the same process. According to the procedure described earlier, the student initially specifies by means of a tag the text sentence from which the passage is to be generated. Two attributes are associated with each tag:

> HDVB identifies the discourse node which is headed by the principal verb of the original sentence.
>
> SCOPE indicates a list of all other verb-headed nodes which were created in the reduction of the sentence.

In the process of deriving the initial sentence, PUSH creates the queue of supplemental sentence-nodes. After displaying the initial sentence, the control function carries on in the manner of POP, except that sentences are derived only from nodes which are defined in the SCOPE list of the specified sentence tag. This device restricts the content of generated passages to that of the designated text segment, while the exhaustive character of the generator assures that every possible sentence root is offered. The user may, in effect, alter the structure of the text by editing the SCOPE definitions for key sentences.

The system permits the use of different grammars for questions and explanations to be designated by the user when the system is initialized. The same grammar is used for both operations in the present system.

Production of Questions

The general procedure used to produce questions was described in the discussion of the tutorial executive subsystem. A question-producing

function is given a sentence produced by the generator and modifies it in the manner described to produce a true-false or fill-the-blank question. In the case of questions, the generator produces a sentence in a special form. As the process was previously described, the user selects *a priori* a list of nonterminal symbols from which a lexical item is derived. This list is ignored during generation of sentences for EXPLAIN, so that sentences are produced as linear strings of words, for example,

(THE SEED IS THE COCONUT OF COMMERCE)

When sentences are produced for QUIZ, the generative function (at the step described above as *gen3*) tests each nonterminal to determine whether it is selected for question-substitution. If it is, then the phrase (always a single word) derived from it is returned as a sublist. For example, in the samples given elsewhere, the nonterminal NOUN was selected for substitution, so that the sentence above would be transmitted to QUIZ* as

(THE (SEED) IS THE (COCONUT) OF (COMMERCE))

To form a question, one of the delimited words is selected at random. Either "-----" or a "falsification-word" chosen at random from the list indicated by the attribute FALS in each lexical entry, is substituted for the selected word, and the resulting string is printed with the delimiters removed, as in

(THE SEED IS THE COCONUT OF -----)

(THE SEED IS THE COCONUT OF FINANCE)

Thus questions are produced by simple string manipulations. This procedure is easily implemented and relatively efficient in operation. Its deficiencies are discussed in the concluding section.

SUMMARY AND CONCLUSIONS

Implementation

The experimental IT1 system was implemented in LISP for the CDC 6600 system at The University of Texas Computation Center. The experimental system is completely interpretive; that is, no functions were hand-coded in assembly language or compiled.

The discourse structure was prepared by hand from a short passage of expository prose adapted from *Compton's Encyclopedia*. The lexicon, which is restricted in content to that necessary for the sample text, was also prepared by hand. A grammar was developed to generate the language of simple sentences required for the application and refined by trial and error until satisfactory results were obtained for the sample text.

Results

The samples of Figure 2 were taken from conversational protocols produced by the author. The reader may compare these sentences with the original text shown in Appendix A. The language generated by IT1 is limited to simple sentences, including adjectival and adverbial modifiers, prepositional phrases, and the limited pronominalization described above. Based on experience with the limited experimental data base, the language seems adequate for the goals of the system. However, certain infelicitous results should be mentioned.

One example appears in the first EXPLAIN passage, generated for S1, in Figure 2. The original sentence was:

> One of the first plants to appear on a newly-formed tropical island is the stately and graceful coconut palm.

The EXPLAIN passage produced by IT1 fails to preserve the meaning of this sentence, which is adorned with several semantic thorns. It is difficult to decompose the sentence into several "simpler" sentences which preserve the original meaning concerning class membership and temporal order of events without resorting to confusing circumlocutions. It is probably inappropriate to decompose S1 into two explanatory sentences, and the discourse structure could be coded to produce essentially the original sentence. Such problems as this are not uncommon in expository prose, but hopefully may be dealt with *ad hoc* without seriously compromising the aims of the system. Automata cannot "explain" everything, and all tutorial systems will continue to include a human instructor to render aid when the automatic components fall short.

The "fill-in-the-blank" questions seem adequate. With the addition of relatively simple answer-processing functions to detect misspellings and similar near misses, quizzes of this type would be quite satisfactory. The true-false questions are less satisfactory. The simple substitution technique used to produce false questions may lead to the infliction of an absurdity such as

The seed is the coconut of music.

or a dilemma such as

The seed is the coconut of industry.

upon the student. To avoid these unfortunate results, the user must exercise a great deal of care in the selection of falsification substitutions to be attached to each lexical entry. This care, applied over the collection of a large lexicon, amounts to a considerable weight of labor. This human effort might be better expended on development of automatic techniques

for selection of falsification substitutions similar to those used by Carbonell (1970).

Computational Resources

ITl, like most natural language processing systems, consumes processor time and storage space at a fairly conspicuous rate. The ITl generator, operating interpretively in LISP, requires an average of approximately 1.2 seconds of central processor time to produce a simple sentence. This experimental system is, of course, quite inefficient by practical standards. The generative grammar is in effect a program which is interpreted by the generative algorithm, which is in turn interpreted by the LISP system. Optimization of the grammar and compilation or hand-coding of the generative algorithm would effect a large reduction in execution time.

Even though the computational efficiency of the generative system would be intolerable in practical applications, it is not prohibitive in experimental work at the present level of utilization. However, the experimental utility of the existing system is seriously compromised by storage requirements. The existing system requires simultaneous central memory residency of ITl, its entire data base, and the LISP system. Experimental progress will be accompanied by increases in the size of the data base, and the limits of the existing conversational system will soon be reached. At a relatively early point in development it will be necessary to convert the system to one in which the data base resides primarily in mass storage, and is drawn pagewise into central memory as required for current command operations.

Applications and Future Development

When considering potential applications for ITl and potentially fruitful lines of development of its successors, it is useful to consider the system in the light of its tutorial functions and data structure. The existing system has two basic tutorial functions: the retrieval of information (EXPLAIN and DEFINE), and the generation of tests (QUIZ). The QUIZ function is, computationally, a special case of EXPLAIN. Questions differ from explanatory sentences in that part of the retrieved information is withheld from the student and is subsequently compared with his response during answer-processing. The scope of the information returned to the student is limited by the command functions by means of the sentence-tags and SCOPE and HDVB properties which bind elements of the discourse network to elements of the original text. These properties impose a secondary structure on the discourse network. This secondary function is used by the command functions to initiate and regulate the generation of sentences.

Thus there are two dimensions along which applications might develop. In one case, the information-retrieval facility might be expanded by implementation of additional command functions which work within the existing structure. An example would be a "tell-about" command, which, given a lexical entry, would return a definition of the entry and a sequence of sentences derived from every discourse node which was related to the entry. It would not be difficult to implement such a command, using the TOK* attribute of lexical entries to initiate a search for prospective sentence-roots in the discourse network.

On the other hand, additional secondary structures might be imposed upon the basic discourse network to serve still other purposes. A

discourse network might be used, for example, as a base for generation of sentences to demonstrate variations of standard grammatical practice, or to produce examples of poor practice to be corrected by the student. For such a purpose, several generative grammars might be stored simultaneously, one of them being selected by the command functions according to their needs. The secondary structure in this case might be one which facilitated selection of discourse nodes of various types, for example, those headed by intransitive verbs, or by transaction-verbs, or the like. A single data base might be used for a variety of instructional purposes. One set of command functions would simply ignore the secondary structures used by the other command functions.

Foreign language instruction seems to be a particularly attractive potential application involving both the basic EXPLAIN function and an extension of secondary structure. The basic ITl system might be very helpful as a reading aid to a student who is beginning to deal with the more complex syntactic constructions of a foreign language. With extension in command and answer-processing functions, the basic generative system might be used to generate drills in verb inflection and the like.

APPENDIX A

ORIGINAL TEXT

(S1) One of the first plants to appear on a newly formed tropical island is the stately and graceful coconut palm.

(S2) The seed is the coconut of commerce.

(S3) It is securely protected from the action of seawater by its thick fibrous husk and hard shell.

(S4) It is peculiarly adapted for distribution by ocean waves and currents, and may be carried a thousand miles from the parent plant to grow on some distant shore.

(S5) This fact accounts for its wide distribution.

(S6) Originally native to certain islands of the Indian Ocean or of tropical America, it is now found on most tropical islands in the East Indies and the West Indies.

(S7) These trees lend distinction and attractiveness to the islands.

(S8) To the cultivation of the coconut is largely due the increasing commerce and rising civilization of many of these remote islands.

REFERENCES

Carbonell, J. R. Mixed instinctive man-computer instructional dialog. Unpublished doctoral dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, June, 1970.

Fillmore, Charles J. Types of lexical information. Ohio State University, 1969 (preprint).

Quillian, M. R. Semantic memory. Unpublished doctoral dissertation, Carnegie Institute of Technology, Pittsburgh, Pa., 1966.

Simmons, R. F. Linguistic analysis of constructed student responses in CAI. TTN-86, Computation Center, The University of Texas at Austin, October, 1968.

Simmons, R. F. Some semantic structures for representing English meanings. Technical Report No. NL-1, National Science Foundation, Grant GJ 509 X, Computer Sciences Department and Computer-Assisted Instruction Laboratory, The University of Texas at Austin, November, 1970a.

Simmons, R. F., & Slocum, Jonathan. Generating English discourse from semantic networks. Technical Report No. NL-3, National Science Foundation, Grant GJ 509 X, Computer Sciences Department and Computer-Assisted Instruction Laboratory, The University of Texas at Austin, November, 1970b.

Simmons, R. L. Compiling semantic networks from English sentences. Unpublished manuscript, Computer Sciences Department, The University of Texas at Austin, 1971.

Zinn, Karl L. Instructional use of interactive computer systems. *Datamation*, September, 1968.