USING RELATIONAL OPERATORS TO STRUCTURE LONG-TERM MEMORY

Alan L. Tharp and Gilbert K. Krulee
The Technological Institute
Northwestern University
Evanston, Illinois

SUMMARY

This paper describes a system for "remembering" a story or passage in English and for the subsequent retrieval of responses to questions. Although it functions as an information retrieval system, it is also intended as a model for long-term human memory. Information is stored in the form of predicates and property lists. In translating from natural language, the system carries out a transformational analysis of each sentence and identifies its deep structure interpretation. This grammatical analysis is essential to the identification of the predicates as well as relationships among them.

The first two subsystems carry out the grammatical analysis. A third identifies the predicates and property lists, determines a time and priority structure, forms a logical map of relationships among predicates, and eliminates some predicates based on an assignment of priority. The fourth subsystem is used to answer inquiries about the stored information.

The method is illustrated with references to one story that has been processed and with examples of questions that might be asked.

Keywords and phrases: Memory, information retrieval, relation, relational operations, predicate, information retrieval, question-answering system.

This paper describes an information retrieval system although it may also be interpreted as a simulation of long-term memory. More specifically, the system simulates the ability to read a story or some form of connected material written in English, to store a representation of that story, and later to be able to answer questions about it based on the ability to recall.

In designing this system, we have been influenced by some related studies which have used an intermediate formalism for the storage.of information. Such systems include BASEBALL , SAD SAM , SIR , DEACON , ALTAIR , SAFARI ,8and the systeras[9]developed by Woods , Rosenbaum , and Hillman . The relative success of these efforts indicates that an intermediate formalism is an important part of the simulation. In addition, each of these systems accepts natural language (English) inputs. The system described in this paper also allows natural language inputs and usee relational operators as an intermediate formalism.

A relational operator connects two entities in a qualitative manner. Two classes of relational operators are used in constructing the model. The first class is denoted e and consists of verbs, which usually join entities (nouns) in natural language passages. The other class, denoted Q> , consists of words that are logical connectives: most of them are conjunctions which, in traditional grammar, connect sentences. The relational operators, g' , bind sentences into larger semantic units.

The first level formalism of a predicate logic is used to represent each sentence of an input passage. The relational operators are the key elements in the predicates, which provide the basic memory structure" for the model. A sentence may be transformed into a two-place predicate which is written e (a B), where a is the relational operator,** is the deep structure subject of the sentence, and 6* is the deep structure object of the sentence. If <t> is null, the sentence is imperative. If *6* is null, the sentence is intransitive. If and B are both null, the predicate represents a participle used as an argument of another predicate.

Sentences joined by the relational operators, g , are treated as n-place predicates, where n is the number of sentences connected by a specific operator. For example, the linked. sentences "S,, and S„ and S " may be represented:

■"[a] 1 S 2 2 3 3 3

Some operators which connect sentences have precedence over others. For example, the relational operator "implication," represented by the logical connectives "if. . .then," is more binding than the relational operator "and."

In translating from natural language into a series of predicates, the system carries out for each sentence a transformational analysis and assigns to each a structural description representing its deep structure interpretation. (See Jacobs and Rosenbaum for a detailed description of transformational theory which includes the concepts of deep and surface structure.) The just tification for this grammatical analysis is essentially as follows. Given a kernel sentence, one can easily translate from English into a predicate. Kernel sentences are closely related to active declarative sentences such that the verb names the predicate and the subject and possibly the object of the verb become the arguments of the predicate. When transformations are applied to kernel sentences, they complicate the relationship between sentence and predicate. For example, the passive transformation reverses the order of the arguments; by using more complex transformations, one creates sentences which may contain two or more kernel sentences. Given a deep structure interpretation of a sentence, one can readily identify the kernel sentences as well as the relationships among them and the process of translating into a series of predicates is facilitated.

The system has been used to process children's non-fiction stories about famous inventions taken from Childcraft11 and makes use of four subsystems. The EDIT component accepts passages which have been manually translated into a restricted subset of English, and these restrictions are necessary because of the limiations inherent in the second subsystem for carrying out the grammatical analysis. (The appendix contains the translated form of the passage "What a Bother.") EDIT replaces pronouns with their corresponding nouns according to the heuristic "replace a pronoun with the most recent noun of the same gender." The pronouns may have referents across sentence boundaries. EDIT then isolates the sentences and forms a map to record the logical relationships among them.

In the second subsystem, Petrick's recognition procedure programmed in LISP for transformational grammars ' is used to assign structural descriptions to the input sentences isolated by EDIT. Before the sentences can be analysed, however, it is necessary to apply manually an inverse number agreement transformation snd an inverse verb-affix transformation. In addition, words such as participles, plurals, and verbs in the past tense have to be separated into their constituent morphemes before the analysis can be performed. Although the grammar chosen for the system was the most extensive one available at the time, it contains many features which limit the class of sentences that it can process. The grammar cannot handle such items as sentence fragments, possessive or reflexive pronouns, most prepositional .phrases and its ability to handle adjectives and adverbial phrases is limited. Clearly, the capabilities of the system could be expanded by the elimination of these snd then restrictions from the grammar.

Within its limits, the recognition procedure assigns to the sentences structural descriptions representing their deep structures.. For example, the sentence, "The store-owner shoved the papers," would be assigned the structural description:

S19(NP(DET(DEF THE))(N STORE-OWNER))
(AUX(AUXA(TNS PAST)))
(VP(V SHOVE) (NP(DET(INDEF S))(N PAPER))).

This LISP S-expresslon corresponds to the tree diagram in figure 1.

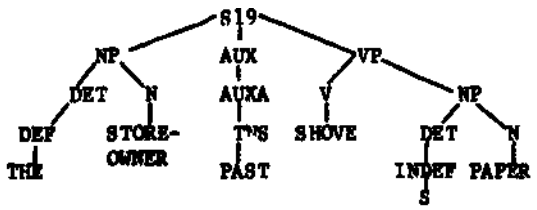

Figure 1

The symbols used as nodes in the tree structure are defined in Table 1. The symbol "S19" specifies that the structure describes the nineteenth sentence in a passage.

S - sentence	NPROP - proper noun
NP - noun phrase	NOM - nominative
AUX - auxiliary	DET - determiner
VP - verb phrase	DEF - definite
TNS - tense	INDEF - indefinite
V - verb (non-copula)	N - noun
BE - verb (copula)	IT - dummy introducing an embedded sentence
NDUM - dummy noun	

Table 1

DIGEST, the system's third component, has two sections. The first segment of the routine forms property lists and predicates snd assigns priorities to the predicates. DIGEST scans the structural descriptions of a sentence, such as the one above, and searches for the verb. If the verb is a copula (BE), the structural description is converted into a property list. If the verb is not a copula (V), the structural description is converted into a predicate. The second segment orders the predicates and then condenses and eliminates information.

To illustrate how DIGEST operates, the structural description of the sentence represented in Figure 1 will be converted into the intermediate formalism. The structural description is decomposed into a noun phrase

(DET(DEF THE))(N STORE-OWNER),

an auxiliary

(AUXA(TNS PAST)),

and a verb phraae

(V SHOVE)(NP(DET(INDEF S))(N PAPER)).

The noun "store-owner" is isolated from the noun phrase snd becomes the first argument of the predicate.

The verb phrase is decomposed into the verb "shove" and the noun phraae (NP(DET(INDEF S))(N PAPER)), The verb is the relational operator which will be used in the formation of the predicate. The noun "paper" is Isolated from the second noun phrase and is labeled as argument two of the predicate. With the second argument isolated, the predicate representation of the sentence is complete:

predicated - shove(store-owner,paper).

Instead of forming a predicate, DIGEST forms a property list if the sentence, such as "The eraser would be handy always," has a copula verb.

The recognition procedure would assign to the sentence the structural description:

S35(N (DET(DEF THE))(N ERASER))(AUX(AUXA(TNS PRES) (M WOULD)))(VP BE(PRED(ADJ HANDY))(ADVB ALWAYS)).

A tree diagram corresponding to this analysis appears in Figure 2,

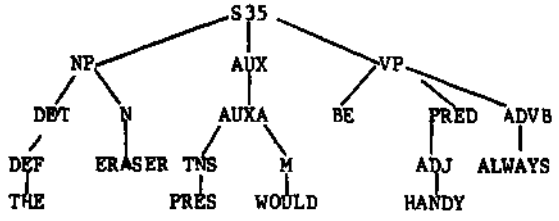

Figure 2

The structural description is again decomposed into the noun phrase

(DET(DEF THE))(N ERASER)),,

the auxiliary

(AUXA(TNS PRES)(M WOULD),

and the verb phrase

BE(PRED(ADJ HANDY))ADVB ALWAYS).

The noun "eraser" is isolated from the noun phrase, and the auxiliary is again ignored.

When the verb phrase with the copula is decomposed, the adjective "handy" is isolated as a member of the class "PRED," and the adverb "always" is also isolated. In forming the property list, both the adjective and the adverb are considered as values of the attributes which they describe. How these attributes (properties) are categorized is explained in the discussion of DIGEST'S lexicon. The adjective "handy" specifies the property of expedience; while the adverb "always" specifies the property of frequency. The noun "eraser," isolated previously, is labeled as the name (head) of the list. The property list formed is

eraser,(expedience,handy)(frequency,always).

The property list is compared with all previously formed property lists, and if an identical list exists, no action is taken. However, if the list is a new one, it is added to the set of property lists for the passage and labeled with the letter "L" followed by an identifying number. The system will process more complicated sentences such as "The store-owner who was Hyman LIpman simplified wirting using a pencil." The structural description for this sentence is given in Figure 3 and from it a suitable series of predicates and property lists will be obtained.

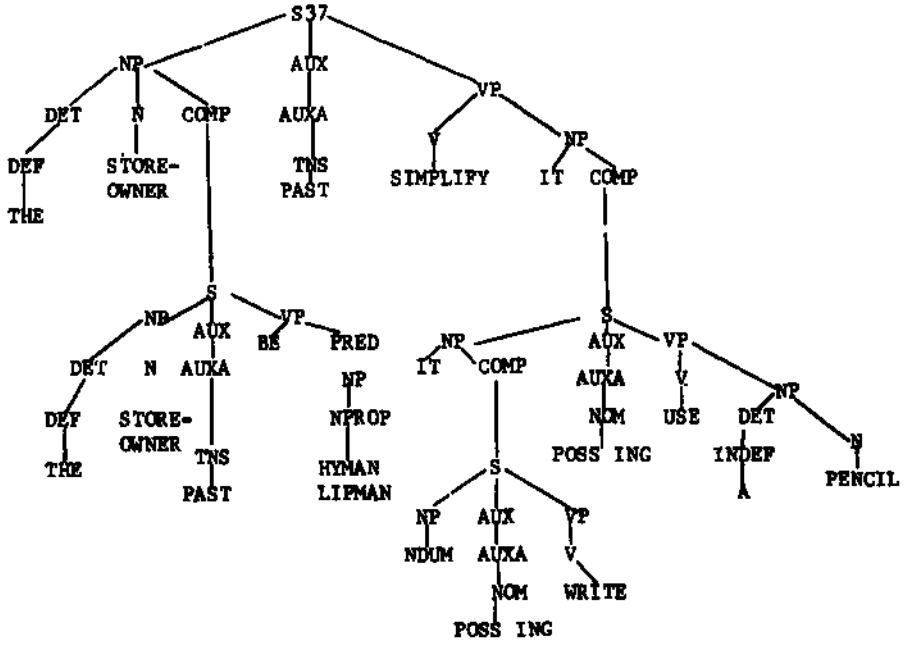

Figure 3

The lexicon (Internalized dictionary) for DIGEST, which is used in the preparatory segment of the routine, has two classes of entries: one for adjectives and adverbs, and one for verbs. The definitions of adverbs and adjectives are used in forming property lists; while the definitions of verbs contain information for assigning priorities to the predicates DIGEST has formed.

The entries for adjectives and adverbs are constructed on a principle similar to that used in constructing a thesarus. A thesarus, rather than defining words, groups them on the basis of similarity into approximately 900 categories. In setting up the lexicon for DIGEST, a more stringent requirement was established: the categories would be mutually exclusive. Since an adjective or adverb may then appear in one and only one category, assigning words to categories tends to be difficult and somewhat arbitrary. The categories chosen are combinations of the categories in Roget's Thesarus For example, the two categories, heat and cold, were combined into one named "heat;" cold is the lack of heat. Each dictionary entry consists of the word, its part of speech (either adjective, ADJ, or adverb, ADV) and its attribute category.

Verbs provide the second type of entry in the lexicon for DIGEST. The entry consists of the verb, a notation which indicates whether it Is transitive (VT) or intransitive (VI), and a priority number which indicates the relative importance of the verbs in a passage. DIGEST uses this priority number along with several other factors In order to provide an Index of the relative Importance of each predicate in forming the memory structure. To the best of our knowledge, no empirical data exlats to support our procedure for the assignment of priorities; it should be interpreted as an educated "guess" about how one might evaluate the relative Importance of each predicate in a given passage. The following factors are used in the assignment of priorities.

(a) Syntactic class of the verb: transitive verbs are more Important than intransitive verbs, which, in turn, are more important than equational verbs (be, become, feel, seem. • .).

[16]

(b) Semantic class of the verb (Kats): activity verbs (eat, speak, walk) are more Important than process verbs (grown, freeze, dress), which in turn are more important than state of being verbs (sleep, wait, suffer).

(c) Semantic sub-class of the verb: verbs of a physical nature (chase, run) are more important than verbs of a mental nature (think, remember) within the same semantic class.

(d) Attache ment of a preposition to the verb: a preposition attached to a verb reduces the verb's transitivity or forcefulness.

(e) Number of meanings of the verb: the information content of a verb is inversely proportional to the number of its menaings.

[18]

(f) Frequency of use of the verb (Kochen) the information content of a verb is inversely proportional to the frequency of its use.

After the individual verb (relational operator) priority numbers have been assigned and stored in the lexicon, they are used in DIGEST'S computation of sentence (predicate) priorities. Three other factors, in addition to that of relational operator priority, are also used in determining predicate priority.

These are the precedence level of the relational operators (if. . .then, and, or, etc.) as previously defined; the location of a sentence in a passage with those at the beginning and end receiving a higher priority and higher priority is given to active sentences over their passive equivalents.

The assignment of predicate priorities, making use of the formula

predicate priority-relational operator(verb) priority

+ relational operator (Q) priority

4- location priority

- correction for passive

completes the first segment of the DIGEST routine.

Condensing the predicates into the most easily remembered information involves several different processes. The predicates are first ordered in a time-priority structure for the initial elimination and condensation. To determine the moat important person of the passage, the predicates are then represented as a connected graph. A centrallty .index as defined by Harary, Norman, and Cartwrlght is computed for each argument. This index is used in order to identify the most important person. The most important person hss the highest centrallty index while the main idea is that predicate with the highest priority. After this determination, the predicates are arranged in a matrix for further elimination and final storage. The time and priority structure Is a linear time axis on which the predicates and their assigned priorities are ordered. Along it, the tenses are ordered as follows: past perfect, simple past, present perfect, present, future perfect, and simple future. The auxiliary part of a sentence, isolated in the first segment of the routine and denoted by the symbol AUX, is interrogated to determine the tense of the verb used in a sentence.*

Once the tense of a verb is determined, the predicate which corresponds to the sentence containing the verb is placed upon one of six tense lists. Each list is push-down: the most recent item added to the list appears at the top. After all the sentences of a passage have been converted into the predicate notation and placed on the tense lists, the lists are joined together in the reverse order of the tense listing above. The resulting ordering along a time axis produces the time and priority structure.

The completed time and priority structure is used twice in the condensation. First it is scanned for predicates with priorities less than one-half the largest priority. Since predicates with such low priorities are evaluated as not important to the main or supporting themes in the passage, they are eliminated. Next, the time and priority structure is checked for identical predicates. Since[1] predicates that occur more often in a passage are more likely to be remembered, the priorities of identical paredlcates are increased.

The predicates which remain after the initial condenstaion are converted into a series of connected graphs, and from these graphs, the most important person, character, or idea in a passage is determined.

For the efficient retrieval, straight-forward revision, or simple addition of a predicate, the graph structure is stored in an n x m matrix, where n equals the number of distinct first arguments of predicates in a passage, and m equals the number of distinct second arguments. The entry for each element of the matrix contains an ordered set of three fields. The first field denotes the name (number) of the predicate; the second, the name of the relational operator (verb) joining the two arguments of the predicate; and the third, the predicate's priority number.

Once the information about the predicate is stored in the matrix, the final condensation can be made. To determine the final priority of the predicates, the most important argument is located. For each predicate that contains this argument, the priority is increased. With the final priorities assigned, another search of the predicates is made, and all predicates with priorities less than one-half of the largest priority are eliminated.

After the final condensation, the following information is then available from the DIGEST routine for use as a data base in the retrieval section of the system:

(1) The matrix containing the information about the predicates,

(2) The set of property lists,

(3) The set of secondary (embedded) predicates,

(4) The time and priority structure,

(5) The logical map formed by EDIT, with predicate and property list names (numbers) substituted for the appropriate sentences.

Using the SNOBOL programming language and operating with a CDC 6400 computer, DIGEST'S analysis of the passage "What a Bother" took 138.4 seconds to execute using 140,000 octal locations of computer central memory. The average time per statement was 8.8 ms.

In the fourth subsystem of the text-based information-retrieval system, information obtained from the data base output of DIGEST is used to answer inquiries about the natural language passages that were inputs to the EDIT routine. This retrieval routine has not been implemented; instead, manual simulations of its functions have been performed.

The routine accepts two kinds of inquiries: questions and imperatives. The questions are inquiries about a single predicate whereas the imperatives are inquiries about the relations of one sentence (predicate) to another in the passage. The questions are parsed according to the same procedure used for the input passages; hence the questions must also be stated in the same restricted subset of English. One advantage of parsing over the keyword matching used in many retrieval systems is that parsing allows a question to be phrased in several ways and yet be mapped into the same predicate. Mapping the questions into the same type of predicate used to represent the input sentences allows a straightforward method of retrieval. Since the imperatives, on the other hand, deal with more than one predicate, the predicate formalism obtained from the parsing output is not useful in answering the second type of inquiry. Hence keywords were used in analyzing imperatives.

Throughout the retrieval segment, inquiries are answered individually. If an answer cannot be determined, the following is printed out: "I do not know the answer to your inquiry (question): could you please rephrase it." If an answer can be located, both answer and inquiry are printed out.

Inquiries seeking information from one predicate fall into two categories: the yes«no question and the WH question. As the name implies, yes-no questions can be answered with either "yes" or "no." WH questions are introduced by an interrogative pronoun (who, what, when, where, which) and are usually answered with a noun phrase. Both categories are subdivided into several question types.

Two types of yes-no questions are differentiated according to whether the verb is a copula or a non-copula. When the verb is not a copula, the question is parsed and transformed into a

predicate. The storage matrix Is searched to determine If the predicate exists. The element of the matrix Indexed by the predicate's two arguments is scanned to discover if it contains the predicate's relational operator. If it does, the answer is "yes;" if not, the answer given is "yes;" if not, the answer given is "no." If an element with the given indices is not found, the "I do not know" answer is printed out.

The other type of yes-no question, that with a copula verb, is also parsed, but it is transformed into a property list Instead of a predicate. The set of property lists is searched to determine if a list with the appropriate name has the attribute-value pair specified. If a property list with the given name is not found, the answer is, "I do not know."

The second major category of questions has six basic types. An interrogative pronoun from one of three functional classes introduces each WH question, which may bgve either a copula or non-copula verb. Sledd , whose classification has been used in the retrieval routine, calls the wh words "interrogative words," rather than pronouns, because they also function *as* adjectives or adverbs. Examples of these three functions include:

(1) pronouns
"Who hit the ball?"
"What did you see?"
"Who is tall?"

(2) adjectives
"Which man is tall?"
"What" man hit the ball?"

(3) adverbs
"When (where) did the man hit the ball?"

A WH question is converted into either a predicate or a property list, and an unknown is sought. If the verb is a non-copula, the question is converted into a predicate with the unknown *as* one of the arguments. The storage matrix is then searched for possible values of the unknown. If the verb is a copula, the question is converted into a property list whose listname is unknown. In this case, the set of property lists is scanned for possible answers.

There are also inquiries which seek information from a passage rather than from an individual sentence in the form of imperatives which seek the semantic relation between the sentences (predicates) of the passage and imperatives which ask about the passage as a whole. A semantic relation is the cause, implication, or other link that builds meaning from two or more sentences. Imperatives seeking such relations are answered primarily with information contained in the logical map (formed in EDIT). A typical inquiry about an entire passage, on the other hand, seeks such information as the main topic or outline. This type of inquiry is answered with information from the storage matrix, especially from the priorities. An imperative Inquiry is handled by locating the keywords in a command. The keyword "give" always introduces an imperative. Representative examples of inquiries and responses for the paasage "What a Bother" are given In the appendix. In addition, deductive capabilities could be added to the system as in the information retrieval system of Green and Raphael Kuhns also has chosen a predicate logic to rorm the data base for his deductive proposal and his method of question answering by computer would use theorem-proving techniques to deduce non-explicit answers from the data base. Such techniques of deduction could also be incorporated into this information-retrieval system, since it also represents sentences with predicates.

APPENDIX

"What a Bother" translated into the restricted subset of English.[1]

The store-owner could not find the eraser. He rummaged through the papers that were scattered on the desk that he had been writing on. He frowned angrily as he shoved around the papers. He became impatient because the eraser was missing.

The store-owner was tired and he was hungry because he had worked all day. He worked in a store that sold stationery. But he could not go to the home until he finished figuring. He kept s record which told how many items were sold each day.

As he finished writing in the book, sll things went wrong. He made mistakes that needed erasing. As he reached for the eraser, it disappeared among papers on the desk. It bounced across the floor.

The store-owner groaned as he picked up the evasive eraser. He looked at the rubber piece he was holding.

He wasted some time as he looked for the eraser. Other people who use pencils must waste some time as they chase erasers. Erasers should be convenient as we are writing.

The store-owner picked up a pencil and he looked at the top. Pencils having erasers that fitted into tops could be made. If a small eraser was glued at the top that a pencil has, then it would be handy always when it was needed.

The store-owner who was Hyman Lipman simplified writing using a pencil.

Original version from the 1968 Edition of Childcraft—The How and Why Library, used with permission of Field Enterprises Educational Corporation.

Inquiries(I) and ResponsesfR) for the passage "What a Bother"

I: Did the store-owner waste time?
R: Yes.

I: Who could not find the eraser?
R: T do not know, . .#

I: Where was the small eraser glued?
R: At the top that a pencil has.

I: Give the main topic of the story.
R-: The store-owner rummaged through the papers that were scattered on the desk that the store-owner has been writing on.

R : A small eraser was glued at the top that a pencil has implies the eraser would be handy always when the eraser was needed.

I: Was the store-owner hungry?
R: Yes.

I: What was evasive?
R: The eraser was evasive.

I: Who kept a record which told how many items were sold each day?
R: The store-owner.

I: Give what implied the eraser would be handy always when the eraser was needed.
R: A small eraser was glued at the top that a pencil has.

I: Which store did the store-owner work in?
R: The store that sold stationery.

I: Where did the store-owner finish writing?
R: I do not know . . .

I: Which people waste time?
R: People who use pencils.

I: Did the store-owner shove around the papers?
R: I do not know . . . #

I: Who wasted time?
R: The store-owner.

I: Where were the papers scattered?
R: On the desk that the store-owner had been writing on.

I: Which eraser is handy?
R: The always eraser is handy.

I: Was the eraser handy never?
R: No.

I: Who was Hyman Lipman?
R: The store-owner was Hyman Lipman.

I: Give the related topics of the story.
R: The store-owner who was Hyman Lipman simplified writing using a pencil.

Note: # attached to an output indicates that a predicate with this information many have been eliminated by DIGEST.

REFERENCES

1. Green, P. F., A. K.Wolf, C. Chomsky, and K. Laugherty. "BASEBALL: An Automatic Question Answerer." Proceedings of the AFIPS Conference. Fall 1961. pp. 133-144.

2. Lindsay, Robert K. "Inferential Memory as the Basis of Machines which Understand Natural Language;" in E. A. Feigenbaum and J. Feldman (eds.) Computers and Thought. New York: McGraw-Hill.

3. Raphael, Bertram. "A Computer Program which 'Understands'." Proceedings of the AFIPS Conference. Fall 1964. pp. 577-589.

4. Craig, J. A., S. C. Berezner, H. C. Carney, and C. R. Longyear. "DEACON: Direct English Access and Control." Proceedings of the AFIPS Conference. Fall 1966. pp. 365-380.

5. Vallee, J.F., Krulee, G.K., and Grau, A.A. "Retrieval Formulae for Inquiry S/sterns." Information Storage and Retrieval. Febr- uary, 1968, pp. 13-26.

6. Walker, D. E., "SAFARI: An On-Line Text-Processing System," Proceedings of the American Documentation Institute, 1967. 4. pp. 144-147.

7. Woods, W.A. "Semantic Interpretation of English Questions on a Structured Data Base." Mathematical Linguistics and Automatic Translation. Harvard Computation Laboratory. NSF Report 17. August, 1966.

8. Rosenbaum, P. S. "A Grammar Base Question-answering Procedure." Communications of the ACM. October, 1967. pp. 630-635.

9. Hillraan, D. J., "Negotiation of Inquiries in an On-line Retrieval System." Information Storage and Retrieval. June, 1968,

10. Jacobs, R.A., and P.S. Rosenbaum. English Transformational Grammar. Waltham, Massachusetts: Blaisdell. 1968.

11. Childcraft—The How and Why Library. Chicago: Field Enterprises Educational Corporation. 1968.

12. Petrick, S. R. A Recognition Procedure for Transformational Grammars. Unpublished Ph.D. Dissertation. M.I.T. June, 1965.

13. Petrick, S.R. A Program for Transformational

Syntactic Analysis. Air Forca Cambridge Research Laboratories Report AFCRL-66-698,

14. Dutch, R.A. (ed.) Roget's Thesarus of English Words and Phrases. London: Longmans, Green, and Co. Ltd. 1962.

15. Kats, J.J., "Recent Issues in Semantic Theory." Foundations of Language. May, 1967, p. 169.

16. Kochen, M., D.M. MacKay, M.E. Maron, M. Scriven, and L. Uhr. "Computers and Comprehension'* in Manfred Kochen (ed.); The Growth of Knowledge. New York: Wiley. 1967. pp. 230-243.

17. Harary, F., Norman, R.Z., and Cartwrtght, D. Structural Models: An Introduction to the Theory of Directed Graphs. New York: Wiley, 1965.

18. Sledd, J. A Short Introduction to English Grammar. Chicago: Scott, Foresman and Company. 1959.

19. Green, C.C. and Raphael, B. "The Use of Theorem-Proving Techniques in Question-Answering Systems." Proceedings of 23rd ACM National Conference. 1968. pp. 169-182.

20. Kuhns, J.L. "Answering Questions by Computer: A Logical Study." RAND Memorandum RM-5428-PR. December, 1967.