AFCRL-66-189

SEMANTIC MEMORY

M. Ross Quillian

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

Contract No. AF19(628)-5065

Project No. 8668

Scientific Report No. 2

October, 1966

(The work reported was supported by the Advanced Research Projects Agency, ARPA Order No. 627, Amendment No. 2, dated 9 March 1965.)

Prepared for:

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

Distribution of this document
is unlimited.

AFCRL-66-189

SEMANTIC MEMORY

M. Ross Quillian

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

Contract No. AF19(628)-5065

Project No. 8668

Scientific Report No. 2

October, 1966

(The work reported was supported by the Advanced Research Projects Agency, ARPA Order No. 627, Amendment No. 2, dated 9 March 1965.)

Prepared for:

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

ACKNOWLEDGMENTS

The author wishes to thank all of the many people who have graciously contributed to this thesis, providing ideas, criticism and research funds. I am especially grateful to my Committee Chairman, Professor Herbert A. Simon, for his continuous support and counsel over the past three years. The other members of the thesis committee, Carnegie Tech Professors Bert F. Green and Allen Newell, and also Professors Daryl J. Bem and Walter R. Reitman (now of the University of Michigan), and Mr. George W. Baylor, all have made many helpful comments on the manuscript at one stage or another. The author is also extremely indebted to Dr. Daniel G. Bobrow of Bolt Beranek and Newman Inc., who has contributed immeasurably to the later stages of the research, and to an old friend, Robert F. Simmons, who has provided both support and encouragement at several stages of the project.

The work has been supported in part by NIH Grant MH 07722, in part by the System Development Corporation, and in part by the Advanced Research Projects Agency, P.R. No. CRI-56176, ARPA Order No. 627, dated 9 March 1965.) The research currently continues under the last named grant.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

page

TABLE OF CONTENTS (Cont.)

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This report describes a model for the general structure of human long term memory. In this model, information about such things as the meanings of words is stored in a complex network, which then displays some of the desirable properties of a human's semantic memory. Most important of these properties is the capability of the memory to be used inferentially; i.e., to allow for the answering of questions besides those specifically anticipated at the time the information is stored in the memory. A computer program is described which illustrates this property by using the memory model inferentially to simulate human performance on a basic semantic task.

When the meaning of some segment of natural language text is represented in the format of the model, relationships and features of this meaning must be made explicit which were not explicit in the text itself. This becomes a methodological advantage in an experiment in which a person reads text and represents its meaning in the model's format, for then certain parts of his otherwise covert "understanding" of the text become externalized, and available for study. A verbal protocol recorded in such an experiment is analyzed. From this analysis a theoretical picture is developed of how text understanding may proceed on the basis of selective interaction between the text and the reader's overall store of prior information.

CHAPTER I

THE ROLE OF SEMANTIC MEMORY

The central question asked in this research has been: what constitutes a reasonable view of how semantic information is organized within a person's memory. In other words, what sort of representational format can permit the "meanings" of words to be stored, so that human-like use of these meanings is possible. In the next chapter an answer to this question is proposed in the form of a complicated, but precisely specified, model of such a memory structure. The test of this model is its ability to shed light on the various types of behavior dependent on semantic memory, preferably both by accounting for known phenomena and by generating new research data. The model's use in explicating various memory-dependent behaviors will be considered.

The first of these memory-dependent tasks is relatively straightforward: to compare and contrast the meanings of two familiar English words. The first half of the dissertation will show that a computer memory containing information organized as the model dictates can provide a reasonable simulation of some aspects of human capability at this task. One program is described that, given pairs of English words, locates relevant semantic information within the model memory, draws inferences on the basis of this, and thereby discovers various relationships between the meanings of the two words. Finally,

it creates English text to express its conclusions. The design principles embodied in the memory model, together with some of the methods used by the program, constitute one theoretical view of how human memory for semantic and perhaps other conceptual material may be represented, organized, and used.

The second behavior investigated in the light of the same theoretical framework is very much more complex: the processing of English text that is done by a person during careful reading, and which will lead that person to report that he has to some extent "understood" the text. The second part of the dissertation is devoted to showing how the representational format and memory model developed and used in the computer program can also serve, first as a methodological innovation to enable collection of new data about the process by which text is understood, and second as part of a theoretical explanation of how that process occurs. This section of the dissertation consists primarily of an analysis of one subject's "thinking aloud" protocol, collected as she performed a complex linguistic task.

A. PRIOR LITERATURE: WHAT IS TO BE STORED IN SEMANTIC MEMORY

Literature relevant to the question of what semantic information is and how it may be stored and used in a person's brain includes a sizable portion of philosophy, a good part of psychology, some of linguistics, and much of the computer programming literature that deals with natural language processing, list processing, or heuristic programs. Rather than attempt to survey all of this now, it will be easier to mention related works at that point in the dissertation where their ideas are either incorporated into, or rejected from the present model. In this chapter, therefore, prior works will be mentioned

only as they help to clarify what the memory model is, or is not, intended to accomplish.

One issue facing the investigator of semantic memory is: exactly what is it about word meanings that is to be considered. First, the memory model here is designed to deal with exactly complementary kinds of meaning to that involved in Osgood's "semantic differential" (Osgood, et al, 1957). While the semantic differential is concerned with people's feelings in regard to words, or the words' possible emotive impact on others, this model is explicitely designed to represent the non-emotive, relatively "objective" part of meaning.

The next relevant distinction is between learning and performance. As a theory, this model does not deal directly with the acquisition of semantic information, but only with what eventually results after a long period of such acquisition. The problem of how humans acquire long-term semantic concepts is simply finessed by having a trained adult (a "coder") build the memory model primarily by hand.

The model is designed to enable representation and storage of any and all of the non-emotive parts of word meanings, of the sort presumably responsible for the fact that a conditioned response to a word generalizes more readily to words close to it in meaning than to words close in sound. (For example, from "style" to "fashion" more readily than from "style" to "stile." Razran, 1939; for a recent survey see Creelman, 1966.) More important, the model seeks to represent the memory that a person continuously calls upon in his everyday language behavior.

The memory most generally involved in language is what one might call "recognition memory," to distinguish it from

"recall memory." For example, if a reader is told that the word "the" can mean "her," he may not immediately recall how this can be so. However, if he encounters text which says, "I took my wife by the hand," he will have no hesitation in recognizing what "the" means. It is this sort of recognition capability, not in general recall, that a store of semantic information must support, and that is the exclusive concern of this paper. Since one ready source of such semantic information is an ordinary dictionary, a coder building this memory model takes much of his information from the dictionary. No less important, however, the coder will at the same time use common knowledge which he himself possesses and must use to read the dictionary material intelligently: the fund of knowledge that constitutes his own semantic memory.

B. SEMANTIC MEMORY IN PSYCHOLOGY AND SIMULATION PROGRAMS

Another historically important issue for memory is what use it makes of associative links. Early philosophical psychology, most current experimental work on "verbal learning," as well as behavioristic accounts of performance such as Skinner's (1957), all make the assumption, to one degree or another, that cognitive and memory structure consists of nothing more than an aggregate of associated elements. At the same time, another tradition and body of work is based on the assumption that attributes and (often) "plans," make up the representational medium in which cognitive processes occur. The notion that attributes (labeled associations) are a key part of the thought medium apparently was first recognized and incorporated into a comprehensive theory by Otto Selz (see de Groot, 1965.) This notion can be found well stated for clinical psychologists by George Kelly (1955), for psychologists concerned with concept formation by Bruner, et al.,

(1956), and in regard to emotive word meanings by Osgood, et al., (1957). The idea that plans form the key part of memory is classically expressed in the work of Bartlett (1932), Piaget (1950), Newell, Shaw and Simon (1958), and Miller, Galanter and Pribram (1960). In this tradition, a "schema" is typically a combination of a plan and denotative data related to that plan. (For attempts to extend some of these approaches and to relate them to computer programs, see Reiss, 1961 (re classical association psychology), Olney, 1962, (re Bartlett), and Quillian, Wortman and Baylor, 1965, (re Piaget.))

It might be felt that the two assumptions above are contradictory, that the cognitive medium must either be associative links, or attributes and plans, (e.g., compare Chomsky's review of Skinner, 1959) However, Newell, Shaw and Simon, attempting to model cognitive processing in a computer, developed a "language" (IPL) in which associative links, attributes, and plans are all representable homogeneously as data. IPL and the later list processing languages provide these, respectively, in the form of lists (items connected by undifferentiated associations), description lists (items connected by labelled associations, thereby forming attributes with values), and routines (equivalent to plans). (For a description of IPL see Newell et al, 1963). By constructing a memory model and program in one of these computer languages, it is possible, taking advantage of the substantial foundation of design and development existing in that language, to use associations, attributes and plans freely as building blocks. Thus, in the programs called BASEBALL (Green, 1961), SAD SAM (Lindsay, 1963), and STUDENT (Bobrow, 1964) the meanings of certain English words were in part stored as factual information, in part as plans. That this same flexibility prevails in human cognitive

structure is also affirmed by sophisticated learning theorists. (See Osgood, 1965).

Therefore, the issue with which a semantic model has to come to grips is not whether to use plans, attributes, or simply associations, but rather what particular sorts of these are to be used to represent word meanings, and exactly how all of them are to be interlinked.

However, while computer programs allow elaborate data structures, very few programs have been much concerned with the structure of long term memory as such. There are exceptions: part of Simmons' "Synthex" project (1963) constituted a thorough exploration of a straightforward approach: namely, a memory consisting of verbatim text (bolstered by a complete word-index). Simmons demonstrated that such a memory can be used to retrieve possibly relevant statements, but not in general to answer questions by inference. Questions formulated within a cognitive orientation different from the one which the input text itself employed are difficult to answer reliably with such a memory. This points up a major goal for a model of semantic memory: the ability to use information input in one frame of reference to answer questions in another frame of reference, or, what is the same thing, to infer from the memory as well as to retrieve parts of it verbatim.

Programs by Green (1961), and by Lindsay (1961), explored the idea of using a memory organized as a single predefined hierarchy. Green's program showed that such a memory can be interrogated with natural language questions, and Lindsay's demonstrated that this kind of memory organization can provide certain inference-making properties, as long as information is confined to a single subject like a family tree. However, this

kind of organization becomes uncomfortably rigid as larger amounts of material are considered, and is clearly not a general enough organization for the diverse knowledge people know and utilize.

Actually, most simulation programs (including those of Green and Lindsay) have not been primarily concerned with long-term memory at all, but rather with cognitive processing. (For surveys of simulation programs see Feigenbaum and Feldman, 1963, and especially Reitman, 1965. See also Minsky, 1961, 1963, Baylor and Simon, 1965, Bobrow, op cit, Raphael, 1964, and Simon and Kotovsky, 1963). One of these programs, Raphael's SIR, creates a small specialized memory from input English sentences, but, again, is not primarily concerned with memory per se. Thus, the problems of what is to be contained in an overall, human-like permanent memory, what format this is to be in, and how this memory is to be organized have not been dealt with in great generality in prior simulation programs. (Reitman's investigations of certain features of such memory structures constitute something of an exception, see Chapter 8, op cit. For a good survey of data bases used in question-answering programs, see Simmons, 1964).

In sum, relatively little work has been done toward simulating really general and large memory structures, especially structures in which newly input symbolic material would typically be put in relation to large quantities of previously stored information about the same kinds of things.

Further advances in simulating problem-solving and game playing (see Reitman, 1965), as well as language performance, will surely require programs that develop and interact with large memories.

C. MEMORY IN LINGUISTIC THEORY

Even more than simulation programs, current linguistic theories have minimized the role of a permanent memory. Transformational, and more generally, all "generation grammar" linguistics analyze language as the application of formal rules. These rules draw minimally on a lexicon (which amounts to a memory for various properties of words.) In Chomsky's recent work (1965, pg. 120 ff.), as well as in the recent thesis of Lakoff (1965), there are several proposals for expanding the role of such a lexicon, and Katz and his co-authors have suggested how a lexicon could be expanded to include semantic information. Also, Lamb (1964, see also Reich, 1965, 1966), allocates one "level" in his "stratificational" view of language to semantic units (sememes), and asserts that these should be discovered by the same procedures that linguists have used to isolate phonemes.

However, in none of these cases has any real effort been made actually to set up a quantity of semantic material and see if it can be used. This is partly because linguists feel that what a person actually does with language is outside their jurisdiction. Chomsky, for instance, specifically divorces his theoretical model from considerations of how people actually deal with language, by insisting that he is modeling a completely abstract linguistic "competence," not the concrete performance of <u>any</u> one person, even an ideal one. Thus, "a generative grammar is not a model for a speaker or a hearer. It attempts to characterize in the most neutral possible terms the knowledge of the language that provides the basis for actual use of language by a speaker-hearer." However, this disclaimer is generally followed by an assertion to the effect that: "No doubt, a reasonable model of language use will incorporate,

as a basic component, the generative grammar that expresses the speaker-hearer's knowledge of the language..."(both quotes are from Chomsky, op cit, pg. 9. For an explicit attempt to clarify the relation of Chomsky's work to actual language performance, see the important paper by Miller and Chomsky, 1963).

Since transformational grammar is a powerful and relatively well-developed body of theory, Chomsky's assertion that such a grammar will be a "basic component" of a "reasonable model" is a strong one, and one that is now generating considerable psycholinguistic research. (Cf. the survey of last year's work at the Harvard Center for Cognitive Studies, 1965, and recent papers such as that of Lane, 1964.)

Assuming a familiarity with Chomsky's theoretical framework, it will be useful to ask how a model of memory should relate to it. The answer depends on whether or not a person's memory for semantic information, as conceived by linguists, is separate from his memory for other sorts of things, such as visually perceived facts, or, in contrast, is part of a general memory which includes these.

If one assumes that semantic memory is strictly limited and separate from other memory, then the former may be allocated to the position expressed by Katz and Postal (1964), who say:

> "The syntactic component is fundamental in the sense that the other two components both operate on its output. That is, the syntactic component is the generative source in the linguistic description. This component generates the abstract formal structures underlying actual sentences...."In such a tripartite theory of linguistic descriptions, certain psychological claims are made about the speaker's capacity to communicate fluently. The fundamental claim is that the fluent

> speaker's ability to use and understand speech involves a basic mechanism that enables him to construct the formal syntactic structures underlying the sentences which these utterances represent, and two subsidiary mechanisms: one that associates a set of meanings with each formal syntactic structure and another that maps such structures into phonetic representations, which are, in turn, the input to his speech apparatus." (pp. 1-2, italics mine.)

Several computer programs have been written that minimize or by-pass the role of semantic knowledge in language. These programs generate sentences that are syntactically grammatical, but whose meanings are either random (Yngve, 1960), or random permutations of the "dependency" constraints imposed by an input text (Klein and Simmons, 1963, and Klein, 1964).

On the other hand, of one assumes that memory for semantic material is no different from memory for any other kind of conceptual material, then this memory must take on a much more important role in language. Here it will be assumed that humans, in using language, draw upon and interact with the same memory in which their non-linguistic information is stored.

Under this assumption semantic memory is simply general memory, and hence must be flexible enough to hold anything that can be stated in language, sensed in perception, or otherwise known and remembered. In particular, this includes facts and assertions as well as just objects and properties.

Under this assumption also, the semantic component becomes the primary factor in language, rather than a "secondary" one subordinate to a separate syntactic component. To consider language production in this light is to put the intended message of the language in control of its format. And, it is to see the reading of text as a continuous interaction between concepts the

text is currently discussing, the reader's general knowledge about the same concepts (part of which has been acquired through nonlinguistic sources), and what has already been stated about those concepts in the same text (or elsewhere by the same author). Making this kind of three-way interaction natural is, therefore, a chief aim of the model to be developed here.

This means that the memory model will correspond less to the proposed semantic lexicons of transformational theory than to what is called "deep structure." Thus, when this memory model is used in a program simulating language production, the program will contain something corresponding to transformational rules, but nothing corresponding to phrase structure rules. The reason for this is that the correspondents to phrase structure rules have been incorporated into the conventions specifying the structure of the memory itself (and there broadened almost to triviality). What remains of such rules would be relevant to a learning program, since this would involve building up new parts of the memory, but is not relevant to a program designed simply to use the memory and to express facts it implies in English text.

In other words, it is being proposed that, in people, language is never torn down into the "immediate constituents" utilized in rules of the familiar S → NP + VP sort. Instead, language is remembered, dealt with in thought, and united to non-linguistic concepts in a form which looks like the result of phrase structure rules - what Chomsky calls the "base phrase marker" or "basis" of a sentence (op cit, Pg. 17). The memory structure will differ from such a basis, or set of them, in that it will not be divided up into small structures each of which is associated with one sentence, but rather will be all one enormous interlinked net. When part of this net is to be

expressed in English text, division into sentences will be made by the text producer as convenient, rather than before this text producer begins to work.

While the memory model to be described corresponds best to the deep structures in transformational theory, it must at the same time serve the role that transformationalists allocate to a lexicon. The same memory structure to which language adds information during intake and from which it retrieves it during output, is also used to interpret language that is read or heard.

The foregoing indicates generally the relation of the semantic memory model to linguistic theory, to other simulation programs, and to some common semantic notions. The next chapter explains the model itself as it is presently formalized in a computer program. The third chapter describes this program and its results. The fourth discusses a method for using the memory model to study how people understand sentences, and introduces a set of data gathered in this way. The fifth, sixth, and seventh analyze these data, again relying primarily on the memory model, and the final chapter considers changes of the model that now seem indicated, as well as some implications of the model for a theory of human memory.

CHAPTER II

THE MEMORY MODEL

A. OVERVIEW OF THE MODEL

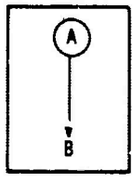
The memory model consists, basically, of a mass of nodes, interconnected by different kinds of associative links. Each node may for the moment be thought of as named by an English word, but by far the most important feature of the model is that a node may be related to the meaning (concept) of its name word in one of two different ways. The first is directly, i.e., its associative links may lead directly into a configuration of other nodes that represents the meaning of its name word. A node that does this is called a type node. In contrast, the second kind of node in the memory refers indirectly to a word concept, by having one special kind of associative link that points to that concept's type node. Such a node is referred to as a token node, or simply token, although this usage implies more than is generally meant by a "token," since, within the memory model, a token is a permanent node. For any one word meaning there can be exactly one and only one type node in the memory, but there will in general be many token nodes for it scattered throughout the memory, each with a pointer to the same unique type node for the concept. To see the reason for postulating both type and token nodes within the memory, it will be useful to reflect briefly on the way words are defined in an ordinary dictionary.

For defining one word, the dictionary builder always utilizes tokens of other words. However, it is not sufficient for the reader of such a dictionary to consider the meaning of the defined word to be simply an unordered aggregation of pointers to the other word concepts used in its definition. The particular configuration of these word concepts is crucial; it both modifies the meanings of the individual word concepts that make up its parts and creates a new gestalt with them, which represents the meaning of the word being defined. In the memory model, ingredients used to build up a concept are represented by the token nodes naming other concepts, while the configurational meaning of the concept is represented by the particular structure of interlinkages connecting those token nodes to each other. It will be useful to think of the configuration of interlinked token nodes which represents a single concept as comprising one plane in the memory. Each and every token node in the entire memory lies in some such plane, and has both its special associative link pointing "out of the plane" to its type node and other associative links pointing on within the plane to other token nodes comprising the configuration. In short, token nodes make it possible for a word's meaning both to be built up from other word meanings as ingredients, and at the same time to modify and recombine these ingredients into a new configuration. Although we will not describe the detailed structure of a plane until Section B, it will be useful for understanding the model's overall organization to examine Fig. 1 at this point.

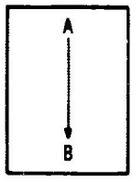
Figure 1-a illustrates the planes of three word concepts, corresponding to three meanings of "plant." The three circled words, "plant," "plant2," and "plant3," placed at the heads (upper left-hand corners) of the three planes, represent type nodes; every other word shown in the Figure 1-a planes represents

Key to Figure 1

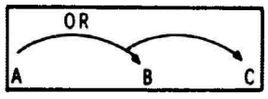
Associative Link (type-to-token, and token-to-token, used within a plane)

1. 

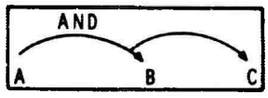
(only where A is a type node) B names a class of which A is a subclass.

2. 

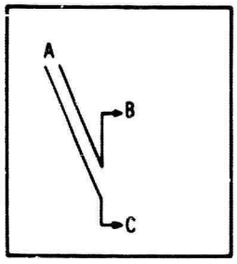
(only where A is a token node) B modifies A.

3. 

A, B, and C form a disjunctive set.

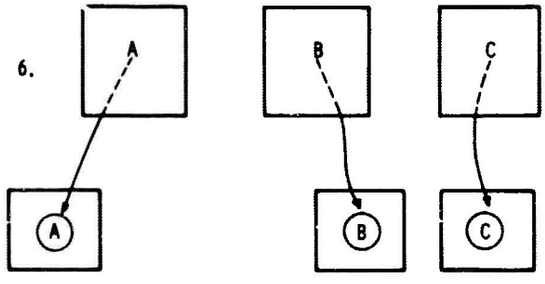
4. 

A, B and C form a conjunctive set.

5. and 6. 

B, a subject, is related to C, an object, in the manner specified by A, the relation. Either the link to B or to C may be omitted in a plane, which implies that A's normal subject or object is to be assumed.

Associative Link (token-to-type, used only between planes)

6. 

A, B, and C are token nodes, for, respectively, A, B, and C.

FIG. 1 SAMPLE PLANES FROM THE MEMORY

PLANT. 1. Living structure which is not an animal, frequently with leaves, getting its food from air, water, earth.

2. Apparatus used for any process in industry.

3. Put (seed, plant, etc.) in earth for growth.

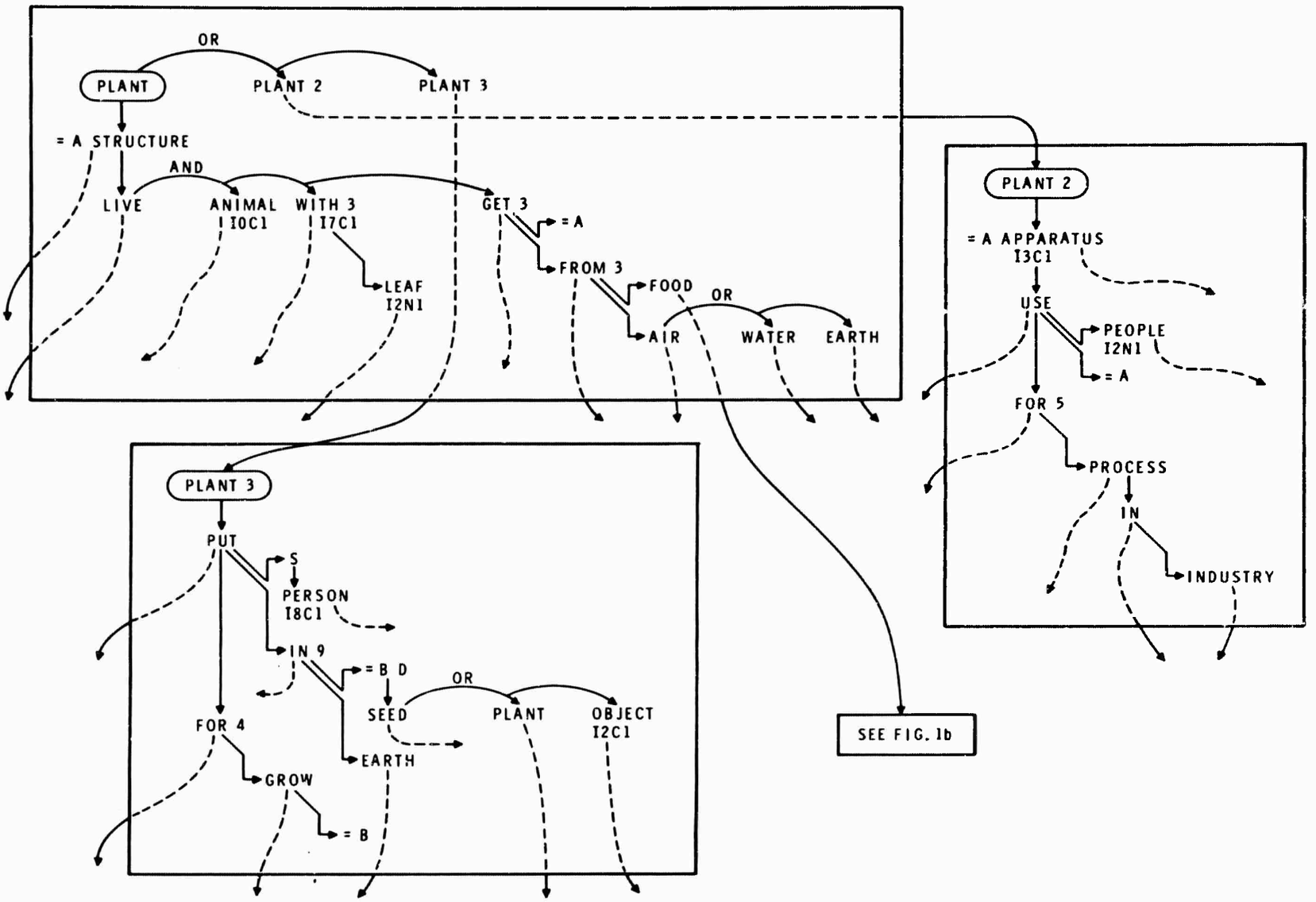


FIG. 1a THREE PLANES REPRESENTING THREE MEANINGS OF "PLANT"

FOOD: 1. That which living being has to take in to keep it living and for growth. Things forming meals, especially other than drink

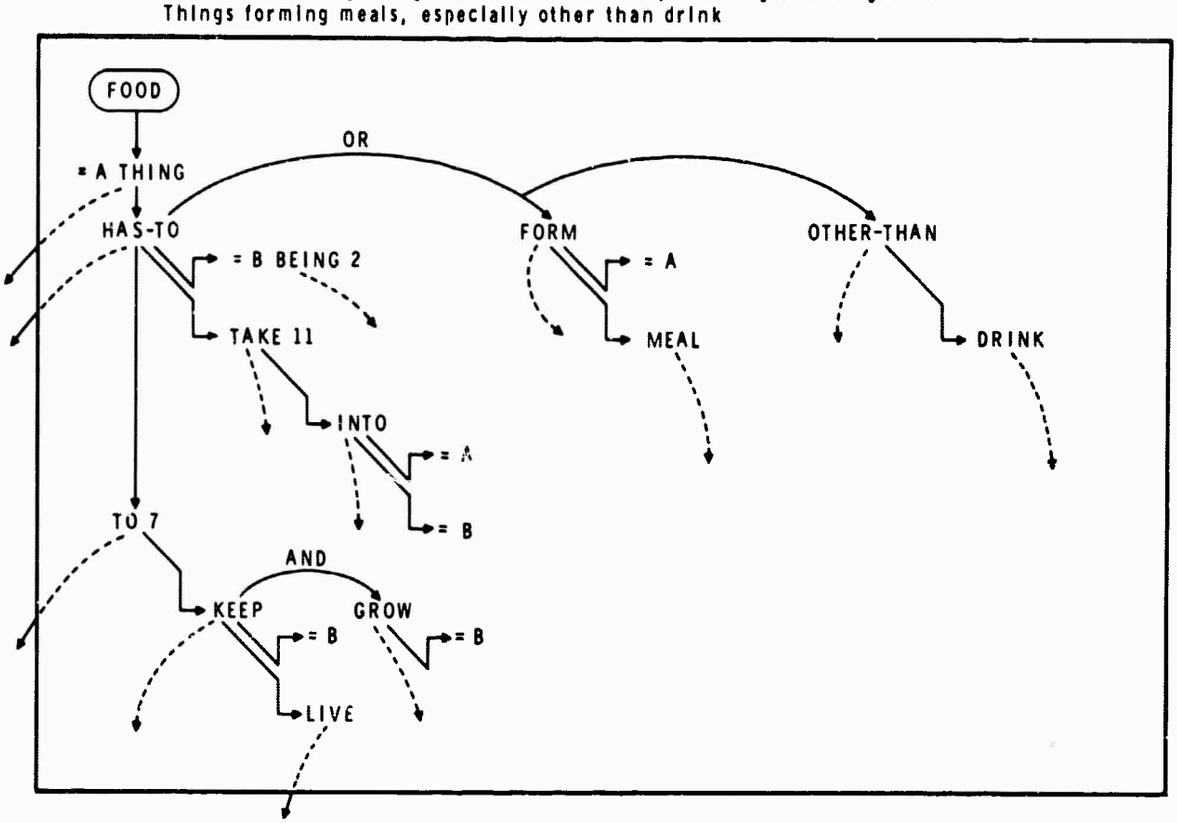


FIG. 1b THE PLANE REPRESENTING "FOOD."

a token node. The nonterminated arrows from tokens indicate that each has its special pointer leading out of its plane to its type definition, i.e., to a type node standing at the head of its own plane somewhere else in the memory. Each of these planes, in turn, is itself entirely made up of tokens, except for the type word which heads it. Figure 1-b illustrates one of these planes. Therefore, the overall structure of the complete memory forms an enormous aggregation of planes, each consisting entirely of token nodes except for its "head" node, which is always a type node.

Now, what is the full content of a word concept in such a memory? Let us define a full word concept, as distinguished from its plane or "immediate definition," so as to include all the type and token nodes one can get to by starting at the initial type node, or patriarch, and moving first within its immediate definition plane to all the token nodes found there, then on "through" to the type nodes named by each of these nodes, then on to all the token nodes in each of their immediate definition planes, and so on until every token and type node that can be reached by this process has been traced through at least once.

Thus one may think of a full concept analogically as consisting of all the information one would have if he looked up what will be called the "patriarch" word in a dictionary, then looked up every word in each of its definitions, then looked up every word found in each of these, and so on, continually branching outward until every word he could reach by this process had been looked up once. However, since a word meaning includes structure as well as ingredients, one must think of the person doing the looking up as also keeping account of all the relationships in which each word he encountered had

been placed by all earlier definitions.

To summarize, a word's full concept is defined in the memory model to be all the nodes that can be reached by an exhaustive tracing process, originating at its initial, patriarchical type node, together with the total sum of relationships among these nodes specified by within-plane, token-to-token links.

Our thesis is that such a memory organization both will be useful in performing semantic tasks, and constitutes a reasonable description of the general organization of human memory for semantic material.

To illustrate the latter point immediately: suppose, for example, that a subject were asked to state everything he knows about the concept "machine." Each statement he makes in answer is recorded, and when he decides he is finished, he is asked to elaborate further on each thing he has said. As he does so, these statements in turn are recorded, and upon his "completion" he is asked if he cannot elaborate further on each of these. In this way the subject clearly can be kept talking for several days, if not months, producing a voluminous body of information. This information will start off with the more "compelling" facts about machines, such as that they are usually man-made, involve moving parts, and so on, and will proceed "down" to less and less inclusive facts, such as the fact that typewriters are machines, and then eventually will get to much more remote information about machines, such as the fact that a typewriter has a stop which prevents its carriage from flying off each time it is returned. We are suggesting that this information can all usefully be viewed as part of the subject's concept of "machine." The order in which such a concept

tends to be brought forth, from general, inclusive facts to obscure or less and less closely related ones, suggests that the information comprising a word concept in the subject's memory is differentially accessible, forming something that may be viewed as a hierarchy beneath the patriarch word. Our memory model's general organization is designed to make a full concept exactly this sort of hierarchically ordered, extensive body of information. The model differs from the memory involved in this example in that we primarily wish to model recognition memory, not recall. Thus, we should actually present the subject with yes-no questions about facts pertaining to machines, rather than have him produce them. However, this could only increase the amount of information involved in a concept and wouldn't change the subject's feeling that some facts are "closer to the top" in the full concept of "machine" than are others.

Clearly a subject has hierarchical concepts similar to that for "machine" for innumerable other word-concepts: "war," "family," "government," etc., so that the overall amount of information in his memory seems almost unlimited. The sheer amount of information involved in such concepts argues strongly that both the human subject's memory, and our model of this, contain as little redundancy as possible, and that it only contain stored facts when these cannot otherwise be generated or inferred. In this regard we note that the information a subject has as the meaning of "machine" will include all the information he has as the meaning of "typewriter," among other things, and there is no need to restate the information constituting his concept of "typewriter" each time it occurs as part of the concept named by some other word, such as "machine," "office," and so on. In short, a word concept like "machine" seems to be made up, in large part, of a particular, ordered arrangement of other word concepts, such as "typewriter," "drill press," etc.

Again, a large memory structured as we have outlined above capitalizes on this redundancy by running the pointer from each and every token node for a word meaning to the same type node. Recall that in such a memory any given type node will have many token nodes, located in various other planes, all pointing to it, and its full concept may well contain token nodes pointing back to the type node that heads one of these planes. In other words, there is no restriction to prevent re-entries or loops within a full concept, so that all routines that search through or process concepts in the memory must take account of these possibilities.

Viewed most abstractly, the memory model forms simply a large, very complex network of nodes and one-way associations between them. Most important, in such a model of semantic memory, there is no predetermined hierarchy of superclasses and subclasses; every word is the patriarch of its own separate hierarchy when some search process starts with it. Similarly, every word lies at various places down within the hierarchies of (i.e., is an ingredient in) a great many other word concepts, when processing starts with them. Moreover, there are no word concepts as such that are "primitive." Everything is simply defined in terms of some ordered configuration of other things in the memory.

A memory organized in this fashion is incomplete, in that other kinds of human information storage and processing — spatio-visual imagery and reasoning, for example — would seem to require other sorts of stored information. It is conceivable that spatio-visual memory is stored in some completely different kind of structure from semantic information. However, it seems at least as reasonable to suppose that a single store of information underlies both "semantic" memory and

"spatio-visual" memory; their difference being not in the structure of the information store, but rather in the way that the static information of that store is used. For example, suppose that a person's visual information is stored in the same interlinked network of nodes that we suggest underlies his language processing, but that he also has the ability to generate visual imagery to directly represent this information, in order to reason spatially. (Cf. Gelernter, 1959). Conceiving of spatial reasoning in this way, with properties abstracted out of actual visual images for purposes of storage, would seem necessary to provide for the flexibility and freedom with which people are able to visually remember, imagine, etc.

Similarly, the ability to recognize objects which are perceived through the senses would require at least some additional kinds of linkages within a general network memory like that we are discussing. But, a network containing one-way associative links from an object's name to the set of properties of that object (as ours does now) would seem already to contain all the nodes needed to recognize a particular object given its sensed properties. What would additionally be required to perform perceptual recognition would be reverse links in the memory, plus a processor able to utilize these links for deciding which object a given stimulus array represented. (cf. Feigenbaum, 1959). A very close interaction between exposure to words and perceptual functioning in people has been thoroughly established (see, for example, Bruner, 1957, Creelman, 1966). Thus, again, it seems logical to suppose that the same static store of information that underlies semantic reasoning may underlie perception, rather than supposing that these rely on separate memory structures, even though, such a memory would then have to be richer in interlinkages than that we shall utilize here.

These and other possible additional functions with a network memory are purely speculative at the present time, and will not be discussed further in the present work. (On a possible relation of the present program to the phenomena of perceptual "set," see Quillian, 1965, pp. 34-36. On the use of spatio-visual imagery in reasoning, see, for example, Paige and Simon, 1966).

B. DETAILS OF THE MEMORY MODEL

Having established the general structure of the memory model as consisting of "planes," each made up of one type node and a number of token nodes, it is further necessary to determine the format of the nodes themselves and the specific varieties of associative links between nodes to be used within a plane.

The most important constraint determining this arises from our assumption that, in order to continue to parallel the properties of human semantic memory, the model must be able to link nodes together into configurations that are at least as varied and rich as the ideas expressed in natural language. Hence, simply attempting to represent natural language definitions accurately in the model becomes a very powerful constraint dictating the model's structural properties. Over a considerable period of attempting to encode English text into such network representations, it has always been found necessary to have available several different kinds of associative links, rather than the simple undifferentiated associations assumed in most classical psychological studies of word association. At the same time, the model must represent all information in a form that is sufficiently standardized to allow processing by rules that can be specified explicitly,

or it will be no more manageable as a theory of memory than is English itself. (See Simmons, 1963, for the most thorough attempt to use English text itself as a computer's store of information on which to base the performance of complex tasks.) The representation now used in the memory model, therefore, lies at a level somewhere between the freedom of English itself and the standardization of, say, symbolic logic. In the memory model, complex configurations of labeled associations must be built up to represent the meaning inherent in dictionary definitions adequately. These are the structures we have called planes.

The attempt to get the meaning of English definitions accurately represented as planes of nodes within the memory model constitutes one major constraint on its structure. A second is provided by the attempt to write programs that can do something interesting by using this memory. To some degree these two constraints on the model balance one another: the first urges elaboration and complexity to represent the meaning of definitions accurately, while the second urges that the model be as simple and standardized as possible to make processing feasible.

As stated above, the relational complexity built up in an English definition is always represented in the memory by a configuration of token nodes linked together to form one "plane." Each token in a plane is linked to its type node (which lies out of the plane) by a kind of association that we show in Figure 1 as a dotted line, while it is related to other token nodes (in the plane) by one or more of the six distinct kinds of associative link listed in the Key to Figure 1. In encoding dictionary definitions, these intra-plane links are used, respectively, as follows:

Link

(1) Dictionary definitions require the use of the subclass-to-superclass pointer whenever they define a word by stating the name of some larger class of which it is a subclass. For example, in the dictionary definition of "plant" shown in Figure 1-a, the word's third meaning is said to be a subclass of the class of "putting."

(2) Any word or phrase used adjectively or adverbially dictates use of the modification pointer.

(3) The multiple meanings of a word, and any phrase such as "air, earth, or water," require the formation of a disjunctive set.

(4) Any phrase like "old, red house" or "old house with a red porch" requires that the modifiers of "house" be formed into a conjunctive set.

(5-6) Together these two links form the open-ended category, by means of which all the remaining kinds of relationships are encoded. This is necessary because in natural language text almost anything can be considered as a relationship, so that there is no way to specify in advance what relationships are to be needed (cf. Raphael, 1964). This means that a memory model must provide a way to take any two tokens and relate them by any third token, which by virtue of this use becomes a relationship.

Stated this way, it appears that the semantic model amounts in structure to a kind of parsing system, and that encoding dictionary definitions into it is in part, at least, similar to parsing these definitions.

This is true, and what appears on one plane of the memory model has many points of correspondence with what Chomsky (1965) calls a "deep structure." In particular, the ternary relations formed by our subject-object links resemble the structure of what used to be called "kernel" sentences. However, our use of terms like "subject," "object," and "modifier" does not always

correspond to that of linguistics, and, also, a plane encodes the meaning of a number of sentences, whereas a deep structure is explicitly limited to the representation of what can be represented in a single sentence (Ibid., p. 138f). Also notice that the correspondence, in as far as it exists, is between one of our planes and one of Chomsky's deep structures, not between a plane and a generative grammar. A generative grammar is an attempt to state explicitly when and how structural information can be related to sentences, whereas the job of a person encoding dictionary definitions into our memory model is simply to get a representation of their structures, i.e., to go ahead and use his language processing abilities, rather than to describe these. Hence our coder does transformations, rather than describing them.

As to the nature of the nodes themselves, it will be assumed that these correspond not in fact to words, nor to sentences, nor to visual pictures; but instead to what we ordinarily call "properties." As indicated earlier this assumption is now common in work on concepts (see, e.g., Hunt, 1962), since properties provide a more elemental and, hence, more flexible medium than visual pictures or words, and since either a mental picture or a language concept may be thought of as some bundle of properties (attribute values) and associations among them.

Thus, the nodes of the memory model actually correspond more to properties than to words, although they may be expressed with words. Representing a property requires the name of something that is variable, an attribute, plus some value or range of values of that attribute. This feature is achieved in the memory model by the fact that every token is considered to have appended to it a specification of its appropriate amount or

intensity in the particular concept being defined. Omitting this specification from a token, which is generally what is done, means that no restriction is placed on the total range of variation in amount of intensity open to the attribute. On the other hand, whenever such specification does appear overtly with a token node, it consists principally of numerical values, stating how the node's total possible range of amount of intensity is restricted. These values allow encoding restrictions to a fineness of nine gradations, i.e., permit nine degrees of "absolute discrimination" to be represented (cf. Miller, 1956). The exact rationale for this kind of specification "tag" has been described elsewhere (Quillian, 1962-b, 1963), along with that of the other two tags (representing the "number" and the "criteriality" of a token; cf. Bruner, et al., 1956) that are available in the model. Here it will only be noted that in encoding dictionary definitions all grammatical inflections, along with all words like "a," "six," "much," "very," "probably," "not," "perhaps," and others of similar meaning, do not become nodes themselves but instead dictate that various range-restricting tags be appended to the token nodes of certain other words. Removing all inflections during encoding permits all nodes in the memory model to represent canonical forms of words, which is of importance both in reducing the model's overall size, and in locating conceptual similarities within it (see Chapter III).

Certain other words besides those mentioned above are also dropped during the encoding process; e.g., "and," "or," "is," "which," "there," and "that," these being interpreted either directly as relationships that are basic structural aspects of the model or else as directions to the coder about how he is to form the plane structure; i.e., as specifications for how the configurations of tokens on a plane are to be structured.

Punctuation similarly shows up only in the associative structure of the model.

All pronouns, as well as all words used to refer again to something mentioned previously in the definition, are replaced in the model by explicit references to the earlier nodes. (In Fig. 1 such referencing is being done by =A and =B, where some higher token node in the plane has been designated to temporarily be A or B by giving it a prefix of =A or =B. A more recent version of the loading program also allows referring to any token node in any plane, by a sort of "indirect addressing" feature.) This ability to, in essence, reuse tokens repeatedly in a plane, perhaps modifying them slightly each time, is extremely important in making the model correspond to human-like memory. In the course of coding a great many words into the current and earlier network representations, I have come to believe that the greatest difference between dictionary entries and the corresponding semantic concepts that people have in their heads is that, while dictionary makers try hard to specify all the distinctions between separate meanings of a word, they make only a very haphazard effort to indicate what these various meanings have in common conceptually. Although they may not be aware of it, there is a very good reason for this seeming oversight: the best the dictionary maker has available for showing common elements of meaning is an outline-like format, in which meanings with something in common are brought together under the same heading. However, as any one who has ever reorganized a paper several times will realize, an outline organization is only adequate for one hierarchical grouping, when in fact the common elements existing between various meanings of a word call for a complex cross classification. That is, the common elements within and between various mean-

ings of a word are many, and any one outline designed to get some of these together under common headings must at the same time necessarily separate other common elements, equally valid from some other point of view. By making the present memory network a general graph, rather than a tree (the network equivalent of an outline), and by setting up tokens as distinct nodes, it becomes possible to loop as many points as necessary back into any single node, and hence in effect to show any and every common element within and between the meaning of a word. The =A notation causes the network building program to create such a link.

In all this, it is clear that not only dictionary definitions but also much of the everyday knowledge of the person doing the coding are being tapped and represented in the memory model being built up. For instance, the reader will already have noticed that a numeral is suffixed to the end of some words (a "1" is to be assumed whenever no such numeral appears). This is simply because it is convenient to have each sense of a word named distinctly within the memory, in order to be able to use these in building other configurations. This means that a person building such configurations for input to the model must always decide which possible sense is intended for every token, and use the appropriate suffix.

C. THE PARAMETER SYMBOLS S, D, AND M

In attempting to encode dictionary definitions it was found that the memory must provide a mechanism for stating that certain nodes in the immediate definition plane of a type node are variable parameters. A value for one of these parameters will be provided only when the word in whose concept the parameter symbol appears is used in text. Other words within that

surrounding text will then form certain parts of the current word's concept; the parameter symbols tell how. To accomplish this, parameter symbols are of three kinds, corresponding to certain ways that other words in text may be related to the word to which the parameter symbols belong. S is the parameter symbol whose value is to be any word related to the present word as its subject; D is the parameter symbol whose value is to be any word related to the present word as its direct object; and M is the parameter symbol whose value is to be any word that the present word directly modifies.

Therefore, to include a parameter symbol in a word's definition plane is to state where within that concept related subjects, objects, and modificands are to be placed, if one or more of these is provided by text in which the present word is used. For example, when the verb "to comb" is defined by the phrase, "to put a comb through (hair), to get in order," this definition is saying that, when used in text, the verb "to comb" is likely to have an object, which is then to be integrated into its meaning in a certain place, vis., as the object of the node "through." In coding the above definition of "to comb," the object parameter symbol, D, would be used as a sort of "slot" to hold a place for this object until "comb" is actually used in text. It is important not to confuse the sense in which D refers to some object of "comb" and the sense in which there are object links within a plane. D always refers to an object of the word in whose defining plane it appears, while its placement in that plane — indicated by the kind of link from some other token node to it — is another matter. For example, notice in Figure 1-a, in the plane for "plant3," the symbol D (which happens also to have been labeled by =B). This D symbol has been placed as the subject of "in9," but it is still a D, because it refers to any direct object of

the verb "to plant." The symbol D specifies that any such object of "plant" is to be integrated into the meaning of "plant3" at the place where the D is placed.

A dictionary definition, in addition to stating where within a concept particular sorts of parameter value information is to be "placed," may offer one or more clue words about what such information is likely to be. Thus, in the definition of "to comb" quoted above we are told that its direct object is likely to be "hair."

Clue words play several roles in the memory model, one of which corresponds approximately to the role transformational linguists ascribe to "selectional restrictions." In other words, the material comprising a full word concept in the memory model can be viewed as consisting of two sorts of information. On the one hand, there is information about the content of the concept itself, on the other there is information about what that concept is likely to combine with when the word is used in text. This latter information is represented by the clue words associated with its parameter symbols. It is significant that this same distinction has been identified in verbal association studies, the associations subjects give to words being divided into paradigmatic (content information), and syntagmatic (parameter clue information) (see, e.g., Deese, 1962). Erwin (1961) has shown that the number of content associations, relative to syntagmatic associations, given by young children steadily increases with age.

In the versions of the memory model used in the programs to be described in this paper, clue words have been sought and coded only reluctantly; both they and the parameter symbols having initially been included only because the sort of information comprising them was embarrassingly present in some

dictionary definitions. However, it turns out that parameter symbols of some kind play a very crucial role in any such memory, because they make it possible to recognize that two different ways of stating the same thing are in fact synonymous. (Examples of this are given in Appendix III, but they will be difficult to follow until S, D, and M, and their related clue words, have been discussed further in Chapters V, VI, and VII.)

As a final point, we note that the model's range readings on tags, together with its ability to form disjunctive sets of attributes, provide it with a ready facility for representing information having a great deal of vagueness. This is essential. It is the very vagueness of the meaning of most language terms that makes them useful — indeed, speech as we know it would be completely impossible if, for instance, one had to specify exactly which machines he had reference to every time he said "machine," and similarly, for every other term whose meaning contains some ambiguity.

To summarize, the memory model, together with the process by which dictionary information is encoded into it, are such that what begins as the English definition of a word seems better viewed after encoding as a complexly-structured bundle of attribute-values — a full concept, as defined above — whose total content typically extends to an enormous size and complexity throughout the memory. Over all, the memory is a complex network of attribute-value nodes and labeled associations between them. These associations create both within-plane and between-plane ties, with several links emanating out from the typical token node, and many links coming into almost every type node.

CHAPTER III

USE OF THE MEMORY MODEL IN A SIMULATION PROGRAM

A. THE TASK OF THE PROGRAM

In selecting a task to perform with a model memory, one thinks first of the ability to understand unfamiliar sentences. It seems reasonable to suppose that people must necessarily understand new sentences by retrieving stored information about the meaning of isolated words and phrases, and then combining and perhaps altering these retrieved word meanings to build up the meanings of sentences. Accordingly, one should be able to take a model of stored semantic knowledge and formulate rules of combination (cf. the "projection rules" of Katz and Postal, 1964) that would describe how sentence meanings get built up from stored word meanings.

It further seems likely that if one could manage to get even a few word meanings adequately encoded and stored in a computer memory, and a workable set of combination rules formalized as a computer program, he could then bootstrap his store of encoded word meanings by having the computer itself "understand" sentences that he had written to constitute the definitions of other single words (Quillian, 1962-b). That is, whenever a new as yet uncoded word could be defined by a sentence using only words whose meanings had already been encoded, then the representation of this sentence's meaning — which

the machine could build by using its previous knowledge together with its combination rules — would be the appropriate representation to add to its memory as the meaning of the new word. Unfortunately, two years of work on this problem led to the conclusion that the task is much too difficult to execute at our present stage of knowledge. The processing that goes on in a person's head when he "understands" a sentence and incorporates its meaning into his memory is very large indeed, practically all of it being done without his conscious knowledge.

As one example, consider the sentence, "After the strike, the president sent him away." One understands this sentence easily, probably without realizing that he has had to look into his stored knowledge of "president" to resolve a multiple meaning of the word "strike." (Consider, e.g., the same sentence with the word "umpire" substituted for "president." Such a decision in favor of one meaning of a word that has more than one possible meaning will hereafter be referred to as "disambiguation" of that word. See, e.g., Rubenstein, 1965.) Just what subconscious processing is involved in unearthing and using the fact that presidents more typically have something to do with labor strikes than with strikes of the baseball variety is by no means obvious, and a good part of this paper is devoted to stating one way that this can be accomplished, given that it has been decided that "president" is the correct word to attend to. Sentence understanding involves a great number of such, at present, poorly understood processes, and the second half of this dissertation will be devoted to developing and using a method of studying how people perform that process, preliminary, we hope, to an eventual simulation program to do so. Meanwhile, the two language functions that the present program performs are considerably humbler than

sentence understanding, although, as will be apparent in the second half, one of them is a crucial part of sentence understanding.

The first of these functions is to compare and contrast two word concepts: given any two words whose meanings are encoded in the memory model, the program must find the more compelling conceptual similarities and contrasts between their meanings. Since, in the usual case, each of the two words to be compared will have several possible meanings, the program is also to specify, for each semantic similarity or contrast it finds, just which meaning of each word is involved. This is one step toward the disambiguation of semantic ambiguity in text. The second major task of the program is to express all the similarities and contrasts found between the two compared words in terms of understandable, though not necessarily grammatically perfect, sentences.

The above tasks are only a part of what apparently is involved in sentence understanding, yet, their performance in a fashion comparable to human performance still calls for a basic degree of semantic horse-sense, in which, heretofore, computers have been conspicuously lacking, and which, apparently, must be based on an extensive and expressively rich store of conceptual knowledge. Thus, being able to get a computer to perform these tasks indicates to some degree the plausibility of the semantic memory model used.

In briefest form, the program that is presently running is used as follows:

1. The experimenter selects a group of words whose definitions are to provide the total store of information

in the memory model during a given series of tests.

2. He looks up each of these words in some ordinary dictionary.

3. He encodes each of the definitions given for each word into the specified format, and loads them into the machine with a program that combines them into a single network of token and type nodes and associative links, the machine's model of a human memory.

4. The experimenter is then free to select arbitrarily any pair of words in the store and to ask the program to compare and contrast the meanings of those two words (requiring that its answers be expressed in sentences).

5. He may then give some fluent speaker the same pair of words, asking him also to compare and contrast them.

6. He compares the sentences the program generates to those the human has produced and, more importantly, considers whether or not the machine's output is one that might reasonably have been produced by a subject.

If the above procedure reveals any changes the experimenter would like to see in the program's performance, he must then revise either some part of the program, or some part of the memory structure or content, or all of these, and test further on new examples to see if the program now operates in a manner closer to what he desires. Repetitions of this kind of test-correct-retest cycle constitute the essence of the simulation method; however, it is important to realize that for the purpose of developing a theory of memory, the result of this development

process should not be thought of as the computer output the program will now produce, but rather as what now may or may not have become clear about the characteristics of workable concept-like memories. Most of the characteristics of which we are aware are incorporated in the model as already described; alterations of this which now seem indicated will be discussed in Chapter VIII.

The present program is designed to compare and contrast the meaning of any two word-concepts in the memory store, and then to generate English text to express each of its findings. Notice that this is not the same task as merely using the two words in sentences, a vastly simpler job, for which one need not even consider the semantic concepts associated with the words (Yngve, 1960).

B. LOCATING INTERSECTION NODES

The actual processing system is made up of three separate programs. The first of these transforms input data (definitions which have been encoded as described in the last section) into IPL form and interlinks these to form the total memory model. This program will not be considered further here. The second program compares and contrasts the two given word concepts. It outputs anything found, but in a form expressed in the memory model's own internal language of nodes and links. The third program takes these findings one at a time, and for each generates English text sufficient to express its meaning. Thus, this third program states (in a sort of "me Tarzan, you Jane" style of English) each similarity or contrast of meaning that the second program has found between the two given words.

It is in the operation of the second program, the comparing and contrasting of two concepts, that the interlocking, token-type structure of the overall memory begins to pay off. For, in order to do this job, it is no longer necessary in such a memory to line up some representation of each of the two concepts side by side and try to compare them. Instead, the entire investigation is simply a matter of searching for points in the memory at which the two full concepts intersect (recall how a full concept was defined in Chapter II above). To see how this is accomplished, recall that the entire memory is a network of nodes and connecting links. Beginning with the two nodes that the program is given to compare (the two patriarch words), this program works alternately on one full word concept and then the other, moving out node by node along the various tokens and types within each. While it will be convenient to visualize this as creating two slowly expanding spheres of activated nodes around each patriarch, actually there is no spatial significance to the expansion of a concept; the nodes in one concept may be located anywhere in the memory model.

The program simulates the gradual activation of each concept outward through the vast proliferation of associations originating from each patriarch, by moving out along these links, tagging each node encountered with a special two-part tag, the "activation tag." Part of this tag always names the patriarch from which the search began, i.e., the name of the concept within which the current node has been reached. Now, the program detects any intersection of meaning between the two concepts simply by asking, every time a node is reached, whether or not it already contains an activation tag naming the other patriarch; i.e., showing that this node has previously been reached in the tracing out of the other

concept. If there is no such tag, the program next checks to see if there is already an activation tag naming the current patriarch; i.e., indicating that this node has been reached previously in tracing out this same concept. If so, the program must take account of this, to inhibit retracing out from the node again and hence repeating its effort, perhaps getting into a loop. Only if neither such tag is found is the node tagged, and further search leading to the nodes it points to considered legitimate.

The second part of each activation tag is the name of the "immediate parent" of the current node, i.e., the node at which the associative link leading directly to it originated. Thus, the "activated" areas of the memory are turned from a one-way network into a two-way network, and, whenever a tag from the opposite patriarch is found, these "immediate parent" parts of activation tags permit the program to trace back "up" from the intersection node to the two patriarchs. This produces two paths, except in the case where the intersection node is one of the patriarchs, in which case only a single path is needed, leading from one patriarch directly to the other.

Examples of such paths and pairs of paths occur in Figures 2-a and 2-b, respectively. The paths from a patriarch to an intersection node produced by the second program should not be confused with the "activation" it makes from each patriarch. While this activation is equivalent to an expanding "sphere," a path is only one particular "line" from the center of the sphere to some point within it, one at which it intersects the other full concept's "sphere."

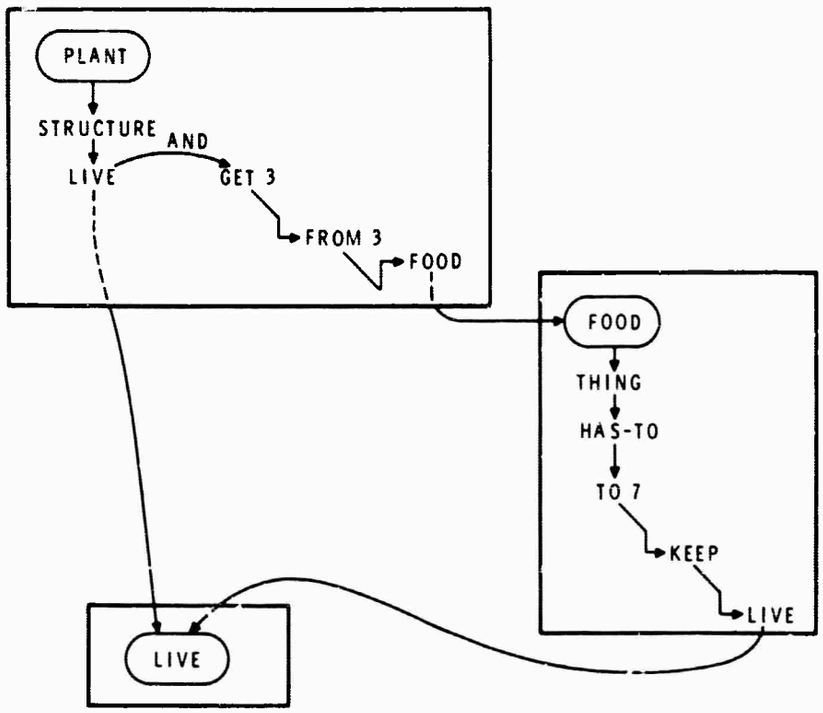


FIG. 2a TWO PATHS DIRECT FROM PLANT TO LIVE

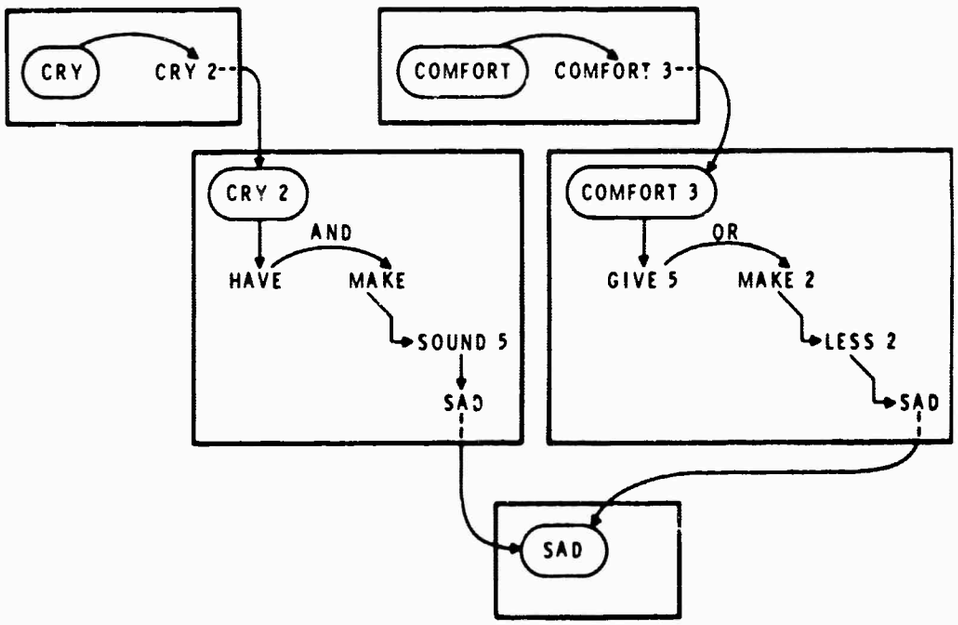


FIG. 2b A PATH FROM "CRY" AND A PATH FROM "COMFORT" WHICH REACH THE SAME (i.e., AN INTERSECTION) NODE.

Expanding the two concepts alternately is extremely important; in effect this makes both concepts into searchers for each other, and gives both the maximal number of targets to look for at any given stage of the search.

C. MAKING INFERENCES AND EXPRESSING FINDINGS IN ENGLISH

The third program, which generates a piece of text to express each path given it by the second program, produces output of the sort illustrated in Table I. (In this table the paths the third program has been given to work on are omitted, although the paths for Examples 1 and 2 are those of Figure 2.)

The most important point about the sentence producer is that there would seem to be considerable justification for considering it, when taken in conjunction with the first two programs, an inference maker rather than just a retriever of information. From a relatively small amount of input data, the overall program will derive a very large number of implicit assertions indeed (see calculations below), and make each such assertion explicit in the form of English text. As an example of its most interesting type of "inferential" behavior to date, the reader's attention is directed to the output shown in Table I as Example 2-B. The path that this output expresses is the longer of those shown in Figure 2-a. As can be seen from a study of Figure 2-b, this kind of performance is made possible by the fact that the memory model interconnects related information which has been input from a great many different definitions, so that, in order to answer some particular question, the search program can trace out a "plane-hopping" path. While path lying completely within one plane (except for its terminal points) amounts only to a representation of some

TABLE I

Example Output from the Current Program
(Paths have been omitted, but see Figure 2)

Example 1. Compare: CRY, COMFORT

A. Intersect: SAD

(1) CRY2 IS AMONG OTHER THINGS TO MAKE A SAD SOUND.[1]
(2) TO COMFORT3 CAN BE TO MAKE2 SOMETHING LESS2 SAD.

(Note that the program has selected particular meanings of "cry" and "comfort" as appropriate for this intersection. The path on which this output is based is shown in Figure 2-b, page 24.)

Example 2. Compare: PLANT, LIVE

A. 1st Intersect: LIVE

(1) PLANT IS A LIVE STRUCTURE.

B. 2nd Intersect: LIVE

(1) PLANT IS STRUCTURE WHICH GET3 FOOD FROM AIR. THIS FOOD IS THING WHICH BEING2 HAS-TO TAKE INTO ITSELF TO7 KEEP LIVE.

(The paths which these two replies express are shown in Figure 2-a, page 23.)

---

[1] "AMONG OTHER THINGS" and "CAN BE" are canned phrases the program inserts when the next thing it is going to mention is one out of a set of things recorded in its memory. At one point, the program was programmed to insert "AMONG OTHER THINGS" whenever it was about to assert one fact out of such a set. We expected this to make its output have a proper, scientifically cautious ring. However, where it had been saying, (rather clodishly, we felt) "TO CRY IS TO MAKE A SAD SOUND," it now said: TO CRY, AMONG OTHER THINGS, IS, AMONG OTHER THINGS, TO MAKE, AMONG OTHER THINGS, A, AMONG OTHER THINGS, SAD SOUND." (!) In short, it turns out that if the program is really made to hedge whenever it knows more than it is going to say, one sits around the console all day waiting for it to get around to saying anything. This may not be such a bad simulation of certain individuals, but wasn't what we had had in mind. Thus, the program is now severely restricted as to just when it can hedge. Science marches on.

TABLE I (Cont'd)

Example 3. Compare: PLANT, MAN

A. 1st Intersect: ANIMAL

(1) PLANT IS NOT A ANIMAL STRUCTURE.
(2) MAN IS ANIMAL.

B. 2nd Intersect: PERSON

(1) TO PLANT3 IS FOR A PERSON SOMEONE TO PUT SOMETHING INTO EARTH.
(2) MAN3 IS PERSON.

(Here the program is treating "person" as an adjective modifier of "someone.")

Example 4. Compare: PLANT, INDUSTRY

A. 1st Intersect: INDUSTRY

(1) PLANT2 IS APPARATUS WHICH PERSON USE FOR5 PROCESS IN INDUSTRY.

Example 5. Compare: EARTH, LIVE

A. 1st Intersect: ANIMAL

(1) EARTH IS PLANET OF7 ANIMAL.
(2) TO LIVE IS TO HAVE EXISTENCE AS7 ANIMAL.

Example 6. Compare: FRIEND, COMFORT

A. 1st Intersect: PERSON

(1) FRIEND IS PERSON.
(2) COMFORT CAN BE WORD TO4 PERSON.

TABLE I (Cont'd)

Example 7. Compare: FIRE, BURN

A. 1st Intersect: BURN

(1) FIRE IS CONDITION WHICH BURN.

B. 2nd Intersect: FIRE

(1) TO BURN2 CAN BE TO DESTROY2 SOMETHING BY4 FIRE.

C. 3rd Intersect: BURN

(1) FIRE IS A FLAME CONDITION. THIS FLAME CAN BE A GAS TONGUE4. THIS GAS IS GAS WHICH BURN.

(The sentence producer starts a new sentence whenever it needs to say something more about something it has used adjectively.)

Example 8. Compare: BUSINESS, COMFORT

A. 1st Intersect: PERSON

(1) BUSINESS5 IS ACT3 WHICH PERSON DO.
(2) COMFORT2 IS CONDITION3 WHICH PERSON HAVE NEED4.

(The code contains information indicating that "person" should be plural here, but the sentence producer does not yet make use of this information.)

B. 2nd Intersect: PERSON

(1) BUSINESS5 IS ACT3 WHICH PERSON DO.
(2) COMFORT CAN BE WORD TO4 PERSON.

Example 9. Compare: MAN, BUSINESS

A. 1st Intersect: PERSON

(1) MAN3 IS PERSON.
(2) BUSINESS CAN BE ACTIVITY WHICH PERSON MUST DO WORK2.

(Something wrong here. I believe a miscoding in the input data.)

B. 2nd Intersect: GROUP

(1) MAN2 IS MAN AS9 GROUP.
(2) BUSINESS2 IS QUESTION3 FOR ATTENTION OF GROUP.

TABLE I (Cont'd)

Example 10. Compare: MAN, LIVE

A. 1st Intersect: ANIMAL

(1) MAN IS ANIMAL.
(2) TO LIVE IS TO HAVE EXISTENCE AS7 ANIMAL.

B. 2nd Intersect: LIVE

(1) MAN IS A LIVE +BEING2.

piece of the information put into the memory, a "plane-hopping" path represents an idea that was implied by, but by no means directly expressed in, the data that were input.

By analogy, suppose we fed a machine "A is greater than B," and "B is greater than C." If then, in answer to the question "what is A greater than?" the machine responded "B," we would not want to call this an inference, but only a "retrieval." However, if it went on to say, "A is also greater than C," then we would say that it had made a simple inference. The kind of path that we have been calling "plane-hopping" is exactly the representation of such an inference, since it combines information input in one definition with that input in another. But the fact that our planes are not simple propositions, but rather sizable configurations, every node of which provides the possibility of branching off to another plane, means that the number of "inferential" paths becomes very large as paths of any appreciable length are considered. Moreover, the possibility that a path may contain fragments from several planes, would seem to clearly indicate that the inferences need not be at all simple, although we do not as yet have actual computer output with which to demonstrate this very conclusively.

Assuming a "complete" semantic memory — one in which every word used in any definition also has a definition encoded — a concept fans out very rapidly from its patriarch. It appears that in such a full memory model the average node would branch to at least three other nodes, considering both its ties to tokens and to its type, if it is itself a token. This means that the average number of paths of, say, up to ten nodes in length emanating from any type node would be over 88,000, each of which would require at least one unique sentence to express.

This is to be compared to 2,046 paths emanating from such a type node if no token-to-type links are available.

Another way to look at the potential of a memory store such as the theory specifies is to compute what the present programs could generate if one could get, say, 850 words' definitions encoded and stored in a memory model. There would then be 360,000 word pairs to ask it about. Since at a conservative estimate a memory model this size would provide ten nontrivial semantic connections, and hence sentences or sentence sets, between the average word pair, the present programs would have the capability to generate well over three-and-one-half million short batches of text to express this total conceptual knowledge, ignoring all that information present only in longer paths. Eight hundred and fifty words' definitions comprise considerably more information than one could model in the core of today's computers (even though an efficient packing scheme might considerably increase the amount one could store). Nevertheless, calculations such as these seem relevant in evaluating the potential of the model as a general theory of long-term conceptual memory.

While a path represents an idea, it is up to the sentence-producing program to get that idea expressed in English. Thus this program must check a path for restriction tags and other features which make it necessary to insert words such as "not" or "among other things" into the sentence generated to express its meaning.

In attempting to express the meaning of a path, this program also deletes, rearranges, and adds words to those given in the path. It works not only with nodes mentioned in the path itself but sometimes looks around these nodes in the

memory model to retrieve additional information and to check on things it considers saying. (For example, in Example 2-b the word "air," although not in the path being expressed, was retrieved to produce legitimate English.)

In expressing a complex path such as that of Figure 2-a, this text-producing program realizes when the capability of its sentence grammar is being exceeded and starts a new sentence. (See, e.g., 7.C.1 of Table I.) Unfortunately, it does this rather often, and a more powerful program clearly would be one which instead of the two sentences shown in Table I as Examples 3.A.1 and 3.A.2 would output the single sentence: "A plant is not an animal but a man is." Some of the minor improvements of this sentence over the two the program now produces would not be difficult to program, but the unification of the two paths into one is a bit more complicated. Clearly, the sentence generation program involves something very close to what Chomsky calls transformations.

In summary, the operation of the sentence producer has little in common with other sentence generation programs, and, in fact, its whole philosophy is contradictory to a good part of the spirit of modern linguistics, inasmuch as this attempts to treat syntactic facts in isolation from semantic ones. As stated earlier, other sentence generation programs produce sentences that, in syntax, are grammatical, but which are in meaning either completely random (Yngve, 1960), or random permutations of the "dependency" constraints imposed by an input text (Klein and Simmons, 1963). The program is also designed in complete contradiction to the subordinate place for semantic information that the formulation of Katz and Postal (quoted on page 9 above) would seem to imply for a performance model. As a theory, the program implies that a

person first has something to say, expressed somehow in his own conceptual terms (which is what a "path" is to the program), and that all his decisions about the syntactic form a generated sentence is to take are then made in the service of this intention. The sentence producer works entirely in this fashion, figuring out grammatical properties of sentences only as these are needed to solve the problem of expressing a path given to it by the search program. (For further details on operation of such a sentence production program, see Appendix IV.)

Thus far, the programs have only been tested on very small memory models, built from no more than 50 or 60 definitions (about 5,000 IPL cells), and on only a few such memories (see Table II below).

A small total memory means that most branches of the proliferating search of a concept are always getting cut short by reaching a type node for which no definition has yet been encoded. One of the most surprising findings from running the program has been that even with this relative paucity of overall information, the program almost always succeeds in finding some intersections of meaning. Actually, Table I lists only a selected sample of the program's output for each compared pair of words; there are usually five or six pairs of sentences generated for each problem pair given to it, although most of these are only trivial variations of a couple of basic sentences such as those we have selected for Table I. The larger the memory model, the greater the number of search branches that remain active, so that the search program becomes able to unearth a great many more semantic connections at a relatively shallow depth beneath any two patriarchs. This ultimately can only improve the program's performance, although it may also require that more concern be given to directing

TABLE II

Words with Definitions Encoded for Use in Model Memories

(Note: Space limitations have so far required that definitions of no more than twenty of these words be used to constitute a model memory during a given series of word comparisons. Since this paper was written, almost all of the 850 words of basic English have been encoded, but not yet run in the program.)

instrument	flame	country	leather
insurance	experience	desire	land
invent	fact	sex	kiss
interest	comfort	plant	know
iron	cloth	family	laugh
ice	cause	meal	light
idea	attack	animal	language
friend	argue	food	law
develop	business	man	lead
event	burn	live	jelly
earth	build	level	journey
exist	bread	lift	jump
drink	behave	letter	judge
fire	cry	learn	

searches than is so far the case. At present, but for one exception, a search just "progressively proliferates" along all possible branches from the two patriarchs (until it has covered a given number of nodes, e.g., 400).

The one exception to this blind, "breadth first," search occurs whenever two concepts are found to intersect on a word used prepositionally, such as "for5" in the concept "plant2." Instead of treating this as a substantive semantic intersection, the search program merely concentrates an immediate burst of search activity out from the two tokens of the preposition. The reasoning here is simply that, while a match on such a word is not in itself sufficient to be treated as a significant conceptual similarity, it is a good bet to examine immediately the subjects, objects, and modifiers of such prepositions, rather than continue the usual search schedule, which normally would not get to these nodes for some time. Unfortunately there is not yet enough evidence available to assess the value of this search heuristic, since its effectiveness, if any, will not show up until the memory model is relatively large.

Another circumstance in which the activation of a concept could be directed is discussed in Chapters VII and VIII.

CHAPTER IV

THE MODEL AS A METHODOLOGICAL TOOL

A. PROCEDURE FOR THE REMAINDER OF THE STUDY

The previous chapter showed that the memory model, once built, can support simulation of a relatively simple type of language behavior. This chapter will propose that the way in which a person takes English text and encodes it into the data format of the memory model is itself worthy of study. This is because, as stated earlier, such encoding of text is not a procedure for which algorithmic rules are available, but, rather, is one that depends upon the coder developing his own understanding of what the text means.

What does it mean to say that a coder "understands" a definition? It seems generally agreed that understanding text includes recognizing the structure of relations between words of the text (as in parsing it), recognizing the referent words of pronouns and of other words used anaphorically, and recognizing the appropriate sense intended for all words with multiple meanings. I take it that the overall effect of these processes is to encode the text's meaning into some form more or less parallel to that in which the subject's general knowledge is stored, so that its meaning may be compared to that knowledge, and perhaps added to it. All this is exactly what a coder has to do to text in order to encode it. Undeniably,

what the coder writes down as his encoding of definitional text will not be a wholly complete or accurate representation of the cognitive material that constitutes his actual understanding of that text. Nevertheless, what he writes down does result from some kind of internal language processing, processing which makes all the decisions mentioned above as elements of "understanding." For lack of a more precise term, therefore, we shall continue to refer to this internal processing that a coder does as his "understanding" the text he encodes.

One might think that this is complicating the problem unnecessarily, that the encoding of text such as dictionary definitions for inclusion in a memory model could be done more simply, without the use of anything as poorly understood as "understanding." Programs such as Bobrow's (1964) have been able to set up the equations corresponding to certain algebra word problems by an almost entirely "syntactic" procedure, and Paige and Simon (1966), studying human solving of the same sort of algebra word problems, found at least some subjects who set up equations for these in a primarily syntactic way.

However, if one attempts to extend the range of language that such a program can handle, it becomes necessary to incorporate increasing numbers of semantic facts. Paige and Simon show how semantic knowledge of various types must be introduced before it is possible to set up the equations for more complex word problems. This semantic knowledge includes assumptions about particular physical situations, about the intentions of the writer of the problem, and certain "conservation assumptions" such as Piaget (1950) has described.

In general, a human introduces and employs such semantic information with little awareness that he is doing so. (For example, Paige and Simon presented their subjects with some problems which, as literally stated, implied physical impossibilities — like a board of negative length. In setting up equations for these problems some subjects simply altered the problems such that they became physically reasonable, without being aware of making such changes. Also, Paige and Simon's subjects often seem to have made use of semantic facts by generating something equivalent to mental images. As stated above this is a step we shall not be concerned with here. See Gelernter, 1960.)

If semantic facts are necessary even for setting up algebra word problems, where most "content" words and phrases can be treated merely as variable names without particular attention to what they refer to, one would think that, in order to interpret English for a less abstract purpose, more use of semantic knowledge might well be necessary. And, in fact, once really sizeable segments of English are considered, the problems, even of parsing sentences correctly, become formidable. Some indication of this is given in Appendix II, which summarizes the results of giving seven sentences to what is probably the best of the automatic parsing programs now working.[1]

These seven sentences were written to correspond as closely as possible to those which we shall deal with below,

---

[1] I am very much indebted to Professor Susumu Kuno of Harvard University for analyzing these sentences automatically with his Multiple-Path Syntactic Analyzer (Kuno, 1963, 1965).

and were not specially chosen to provide particular difficulties for a parsing program. (In fact, they all were adapted from an entry in the Basic English Dictionary, which is supposed to employ only a very simple grammar.) Nevertheless, Kuno's program produced 120 parsings for one sentence, and 1,066 for another.[1]

One cannot sweep all these parsings under the rug as just "linguistic addities." Many of them are, and in fact all parsings save one or two are, in any real appearance of a sentence in text, "oddities." But, saying this merely begs the question, which is: how to decide which parsing is appropriate in each given case where the sentence appears.[2] It should

---

[1] This gives some indication of the amount of ambiguity that remains after unaided syntactic analysis, although even this grossly underestimates the situation. (See note in Appendix II).

---

[2] Professor Simon has raised the following objection: The clearest way to determine whether such sentences can be coded by purely syntactic means, and without use of meaning is to replace all but the syntactic "function" words by nonsense syllables and then test whether human subjects can code them unambiguously. Existing mechanical analysis schemes, even the best, may fall far short of using all the syntactic cues that are available to humans, hence may underestimate the extent of the syntactic information that is present. A preliminary study by the method Professor Simon suggests indicates that human performance with syntactic clues alone may, indeed, be better than a parsing program's performance might lead us to believe. An interesting project would be to determine exactly the extent to which parsings so obtained would be correct for representative samples of various kinds of naturally occurring text.

be noticed in Appendix II that parsings increase multiplicatively, so that it takes only three two-way ambiguities and one three-way ambiguity to generate 24 separate parsings of sentence number 2.

Therefore, resolving just one of the 2-way ambiguities would in this case eliminate half of the total parsings for the sentence. Although how to eliminate these ambiguities automatically is not known (but see Chapter VI below), it certainly would appear that they should, as much as possible, be resolved before any parser labours to generate them.

One might conceivably argue that the fact that no one has yet been able to develop a completely successful parser, or an anaphora recognizer, or a word disambiguation program, does not prove that someone may not in the future do so, and do so without using any semantic information in his programs. However, I think few of those now in the field would believe that this is likely.

In view of the above, it seems to me that any general program able to encode the text of dictionary definitions for inclusion in a memory model is going to have to develop something quite close to what we call an "understanding" of such text. To do this, the program will have to integrate its use of syntactic and semantic facts, rather than complete one type of analysis before beginning the other.

This chapter, then, will consider the use of the encoding process to collect data about a very complex type of human

behavior, namely, the processing of English text people do during reading. The aim of the remaining chapters will be to discover as much as possible about exactly how this processing can be explained.

Ideally, our explanation of how this processing occurs would be so explicit that we could state it as a computer program that would simulate the process. We will not be able to achieve this level of success. Understanding text involves a great deal of information processing. The Harvard Center for Cognitive Studies Report for 1965, summarizing a year of experimental research on language processing, states:

> Taken all together, such investigations have given considerable credibility to the claim that the way people perceive, remember, and understand sentences cannot be explained...(with any simple model). Devising a performance model that does all this - and in a single pass from left to right in real time - will be no easy task." (page 20)

This paper will merely attempt to lay some of the groundwork for an eventual performance model of text understanding and remembering, by analyzing and interpreting data gathered as a coder encodes text.

This analysis will proceed by assuming that the coder's own internal memory is of the same general form as that of our model. At this point the procedure begins to sound, in the apt term of Professor Newell, incestuous. Isn't there some logical circularity involved in collecting data from a subject as he encodes text into the notation of a memory model, and then analyzing his performance by assuming that his own memory is like that of the model itself? This suspicion grows when

we note, further, that our coder-subject has been very carefully trained in the coding procedure. It will be necessary to consider the logic of this procedure very carefully.

First, what a coder is taught is only a notation with which to express his understanding of some particular text, not how to attain that understanding. Specifically, the coder is presented with the links available in the model, as enumerated in the Key to Figure 1. Then he is shown (with examples like Figures 1-a and 1-b, and an explanation like that on page 25) how the meaning of text may be represented by writing down the words of the text, appropriately interconnected by the available links. Then the new coder is given other text to encode into the model's notation. (The coder does not put in the links shown as dotted lines in Figure 1, since these are added automatically by the loading program.) For the first few hours the new coder will make many errors. Some of these are trivial, involving such things as spacing correctly in order to provide acceptable input for the loading program. Other errors involve a misrepresentation of what the coder intends. (The instructor's comments on this early coding often take the form, "Well, what you've represented here means that X modifies Y, is that what you meant to say?" To which the student coder may either assent, or reply something like, "Oh, no, I meant that X modifies Z, so.....this link should have been like this, right?") As such drill proceeds, the number of errors by the new coder declines, until he learns to use the notation accurately to say what he means.

Teaching someone the encoding process thus is like teaching a very good chess player a new notation with which he can express, say, the chess moves he considers in the course of analyzing some position. This notation will not determine how the player analyzes any particular position, nor does our encoding procedure determine how our subject understands any particular stretch of text. It is true that teaching anyone a language to describe some action will affect the way he performs that action, but, in this case, even more than in that of the chess player, understanding of text is a process that an adult has so thoroughly overlearned that it is doubtful that our providing him with a representation can affect this process very significantly. Our faith in this assertion is strengthened by the fact that at some points we have tried to get coders to make certain discriminations, or represent things in certain ways, which they have resisted mightily. The coder would usually acquiesce temporarily, but very soon go back to the kind of procedure he found natural; several clear instances of this will be pointed out in the next two chapters.

Second, during encoding, all the subject is given to work on and with is the stretch of text defining a word. This has been taken from a dictionary. Part A of Appendix I shows the data given to the subject in the case we will be primarily concerned with for the remainder of this paper. This subject, AX, had never previously seen this text. Part B of Appendix I shows what she then produced as her encoded representation of what she understood Part A to mean.

In order to study the process by which AX produced Part B from Part A, a "thinking aloud" protocol of her thought process was collected as she performed the encoding. (See deGroot, 1965, and Newell and Simon, 1964, on this methodology.)

Part C of Appendix I shows this verbal protocol that was given by subject AX as she did this encoding, along with (down the right half of the page) our categorization of the steps she took, and a running commentary on her coding. At the beginning of the protocol, what is now Part B, the encoded plane, was just a blank page; one can trace in the protocol the successive building up of this plane. (Only by referring back and forth between Parts B and C in this way will the protocol be intelligible.) We will discuss Part D of Appendix I, the categorization of steps, in Chapter 6.

Now, in order to analyze and interpret this protocol, we will assume that AX's own internal memory is of a certain overall format, and that she uses this memory in certain ways to understand and encode text. Is this logically circular? The answer is that it is not, exactly to the degree that AX's encoding process involves general text "understanding," or some other internal information processing, that is above and beyond what we have taught her. We believe it will be obvious to anyone who looks closely at the encoding process that it does depend completely on the subject's forming some thorough understanding of the text, and then making a great part of this understanding explicit. Note that this understanding need not be "correct" in any strict sense, only that it must exist. Building a plane is a continuous series of decisions, quite clearly dependent upon the subject forming some detailed conception of what the text means.

To see just how a circularity could result during such a procedure, suppose we had provided the subject with some store of data, or had given her some algorithm specifying how she was to search through and process either this data or her own memory in order to create her coding output, Part B. Under

such circumstances, clearly we would to a large extent be determining her output, and any explanation of it in terms of that data or algorithm would be non-informative. However, none of this was in fact done, AX was given only the text, Part A, and told to encode its meaning as she understood it. At no point in her training had any attempt been made to teach her specific semantic information, nor was she taught any memory searching algorithm. To the degree that we are correct in thinking that the subject is carrying out independent processing of the definitional text when she encodes it, it seems clear that no assumption by us about that independent processing, or the internal memory it depends upon, can have determined it. In particular, this includes even the assumption that her internal memory contains interconnections and links of the same sort as those she has been taught to use in representing the text's meaning. In fact, to the extent that, during encoding, our subject is performing any information processing which is independent of our teaching, it is at least logically non-circular to assume that she does this by employing a memory like that of our model.

B. ADVANTAGES AND DISADVANTAGES OF STUDYING UNDERSTANDING BY STUDYING THE ENCODING PROCESS

Very little is known about how any of the processes constituting text understanding actually are accomplished, although considerable amounts of effort have been devoted to making computers perform certain parts of this processing, especially the parsing of sentences. However, even for this relatively limited task, existing programs are by no means able to match human performance. The best sentence parsing programs come up with a great many parsings for even relatively simple sentences, and may fail completely on sentences of no greater

complexity than one customarily finds in scientific writing. A non-simulation approach, studying directly the way that people understand sentences, encounters formidable methodological difficulties. This is because the process by which a person understands sentences operates very fast, and is not accessible to the consciousness of the subject himself. A person knows what a sentence means, but not how he knows what one means. The current report of the Center for Cognitive Studies of Harvard (1965) states:

> "To 'explain' speech perception we must propose a device whose input would be the acoustic speech signal and whose output would be the meaning that native speakers retrieve from that acoustic signal. Without a satisfactory semantic theory, we cannot even specify the output of such a device." (Page 16, italics mine.)

The methodological importance of a semantic memory model for studying understanding stems from a fact that it does, under certain circumstances, provide a way to make a reader's "output" not only specifiable, but visible. That is, a coder who is encoding is taking English text as "input" and then giving as "output" a plane, to represent the text's meaning. In this plane a great many of the direct results of his process of understanding the text are represented overtly. In particular, in such a plane the results of the coder's parsing of the text, of his disambiguation of its words' multiple meanings, and of his identification of its anaphoric references, can all be identified. For the researcher this means that he is able to observe, for a subject performing a process very much like "understanding," not only an input — the text —, but also an output — the representation the subject builds to represent its meaning. To my knowledge, there is simply no other representation existing which permits this in anything like a

comprehensive, economical manner.

While the encoding process is, of course, not identical to the covert processing which constitutes the understanding of the same text during normal reading, it is very heavily dependent upon such understanding, and is in some ways a slowed down, overt version of it. And, it is precisely such a slowed down version that is needed in order to investigate the understanding process. Having this slowed down version, we can study the encoding process in the same way that other cognitive processes, such as playing chess, have been studied, and apply part of what has been learned from those analyses to the problem of text understanding.

Unfortunately, becoming really accustomed to using the encoding scheme does take considerable practice, so that the protocol of another subject, GB, who had had only about four hours of coding training, is primarily concerned with his figuring out how to represent in our notation what he gets from understanding the text. He eventually does an excellent job of this, and encodes the text more like this writer would than does AX, an experienced coder. But, there is so much of this other subject's protocol concerned with notation that it cannot be very close to his normal understanding process. In AX's encoding, how to represent something she knows has become almost automatic, so her protocol lies much closer to the process we are interested in. As a general method of studying text understanding, our procedure is, therefore, considerably limited by the difficulty of inducing a subject to encode words steadily for a week or more, yet this is essential to get his notational habits to become sufficiently automatic so that the process of primary interest will be more directly visible in

his protocols. We were very fortunate to have one such subject.[1]

Of course, to be able to see the subject "understanding," even in a slowed down form, is not automatically to know how she is proceeding. It will still be necessary to make a very detailed analysis of the protocol in order to, in a phrase of Newell and Simon's (1964), "lay bare the reasoning the subject employs" while she does the processing the protocol traces.

While we will not be able to produce a general simulation program to simulate this processing, we will try to answer one of the central questions about how people understand sentences, which would seem crucial to the design of any program designed to do this. The question is this:

> As the subject processes text to produce an understanding of its meaning (a plane, in our case), just when and how does the processor he uses rely on syntactic information, and when and how on semantic information?

In other words, at which steps can the operation of this processor best be characterized by rules whose conditions for applicability are stated in terms of such things as word order;

---

[1] Subject AX is 22 years old, with two years of undergraduate credit in mathematics at a state university, and one year of credit in psychology. In her second year, her grades were poor (apparently in part because of outside problems), and she switched to extension school and into psychology. She was working full time at the System Development Corporation during the summer in which the coding was done, and had over a month of full time experience coding when the protocol was collected. Her off-work activities involve little reading.

and at which steps does one need rules which utilize a general knowledge of the world, or of concepts previously established in the same text, or of both. A major aim of our analysis will be to sort out the operation of these two sorts of processes within the overall "understanding" processor, which will require that we formulate as explicitly as possible the specific activities involved in both types.

Along with this question two others should be mentioned, which are prominent in current linguistics and psycholinguistics, but which are not of concern to us. The first of these is whether or not pieces of text that an understander processes in fact constitute grammatical sentences. (Chomsky, 1957). The second related question is whether or not a given sentence is "anomolous."[1] The reasons we do not consider these questions very fruitful for psychological research or for artificial intelligence will become evident below.

In all, approximately four hours of protocol data for each of two subjects has been recorded on tape, transcribed, and investigated, although here only the protocol of subject AX will be considered. All of this was collected as follows:

First, the author preprocessed the dictionary definition text as he always did before giving it to a coder. This processing consists simply of lumping together definitions which the dictionary separates, but which seem to differ only slightly, or only grammatically, and of deleting very obsolete or uncommon meanings. Part A of Appendix I shows how this was done,

---

[1] Roughly, an "anomolous" sentence is defined as one which, although syntactically correct, cannot be understood except perhaps metaphorically. For example, "the paint is silent." See Katz and Foder, 1963.

Meanings 2 and 8 being deleted and old Meanings 4, 5, and 6 being lumped with Meanings 1, 2 and 3, respectively. Note that an error was made in lumping 4 with 1, it would be better as a part of 3. Subject GB, given this text to encode, noticed the error, subject AX did not. The text shown in Part A of Appendix I is all that was given to the subjects to work with.

Each subject was told to encode the meaning of that text into the notation they had been taught and, as they did so, to state into the tape recorder as nearly as possible all of the mental process that he went through in doing so. "Whip's" defining text was the fourth definition that each subject had encoded aloud, so that they were very probably beyond any distraction due to the microphone. After all coding was completed, AX herself transcribed her own protocols. For "whip" she was told to divide the continuous flow of dialogue into numbered paragraphs such that approximately one "step" occurred in each paragraph; the actual decisions as to what constituted a step were left to her. Following the methodological strategy advocated by deGroot (1965), AX was also asked to edit her protocol by adding additional comments or punctuation, but not to remove or change anything. Such additions appear in parentheses in Part C of Appendix I. Her work was checked against the tape. During editing, AX herself suggested the underlining that occurs in her protocol, indicating that a word means the word as such, rather than its referent. She also put in the quotation marks, and all punctuation.

In summary, it should be understood clearly just what may and what may not be expected from the approach taken in the rest of this paper. What may not be expected is that we will <u>prove</u> anything about how people encode text or understand

sentences. The most that might be achieved would be an insight into the processing techniques or strategies which they use. Corroboration will have to come later, either by showing that a program using these strategies can effectively recreate the process and itself "understand" text, or by specific experimentation with human subjects, or by both.

Also, if the encoding process does indeed do some degree parallel the normal process of understanding text, and if we can characterize this process by some set of specific rules, then these rules may also be the ones involved generally in people's understanding of text. If coding is not at all like ordinary understanding of text, then the rest of this paper is merely a consideration of how one might get a computer alone to build the model memory from English text. As a study of understanding the only justification for such an oblique approach is the extreme difficulty of studying the same process by other means, and the paucity of plausible assertions about the understanding process up to now.

CHAPTER V

A THEORY OF TEXT UNDERSTANDING VIA SEMANTIC MEMORY

A. UNDERSTANDING AS COGNITIVE PLANE BUILDING DIRECTED BY MEMORY

In order to exploit the protocol data most effectively to add to our understanding of how people understand text, it will be useful to have some overall theoretical view of what understanding text is. At present the only approach to such an overall theoretical view is the "meta-theory" of "semantic interpretation," growing out of the work of Chomsky and his associates on transformational grammar. (See Katz and Fodor, 1963, Katz and Postal, 1964, Miller and Chomsky, 1964, and Chomsky, 1965). However, the general hypothesis of this paper — that a person's memory is of the same general form as that of our model — also leads to a conception of what must be happening when a person reads and understands text. This conception is at some points similar to, at some points very different from the proposals of the transformationalists. The main purpose of this chapter is to formulate this tentative overall picture of how text is understood, based on the assumption that an understander has a semantic memory functionally similar to our model.

The theoretical view here draws very heavily on the sort of memory and memory use embodied in the program of Chapter III, but at the same time requires hypothesizing several other ways of processing such a memory. A second purpose of this chapter, then, is to specify some of the additional processing routines that would seem necessary to a conception of the text under-

standing process. Although this theory of understanding derives in large part from the protocol analyses of the next two chapters, it is placed here so that the overall theoretical picture behind those more detailed analyses will be clear.

We assume that the "meaning" of text is always some (old or new) association of concepts. Thus, to understand such meaning is either to find or to create in the brain of the understander some configuration of symbols, (token nodes) linked together so as to show how certain concepts are associated by the text.[1] Our theory, then, asserts the following: the cognitive processing which a reader must carry out in order to build such a "plane of token nodes" - whether just fleetingly or as a lasting addition to his memory - is based on his finding, for certain pairs or trios of concepts which the text associates, some way in which those same concepts previously have been, or intelligibly may be, associated, given his general memory. In other words, elemental to the understanding of text is the kind of task performed by the program of Chapter III. This is because finding some path in memory connecting two given word concepts, as that program does, amounts to relating those concepts on the basis of the reader's store of prior knowledge, and such relating is the key step in understanding text which relates those word concepts.

---

[1] Actually, one would suppose that a reader would often find, already stored in his memory, planes of token nodes that represent the meaning of whole phrases, sentences and larger units of a text he is reading. He could take advantage of this by representing the currently read text's meaning with an abbreviated plane of higher level tokens, which, by pointing to already stored configurations, could avoid re-representing their information in complete detail. Such "chunking" operations have been postulated to explain many sorts of cognitive behavior. (See, e.g., Miller, 1956, Melton, 1960, and Simon and Feigenbaum, 1959.) However, for purposes of simplicity here we shall ignore this higher level chunking and talk as though a plane of tokens is always constructed to represent the meaning of text in the way that, for example, Figures 1-a and 1-b represent the meaning of the definitional text quoted with them.

A reader is viewed as continuously selecting from text pairs (or slightly larger aggregates) of word concepts to be taken as "patriarchs," so that their interrelationship in memory — their first connecting path(s) — can be located. For text he understands easily these paths will be short. In other instances, he may find in his memory only some very long and indirect path between the associated concepts. This nevertheless means that their association is, by means of that long path, "intelligible," within the overall frame of his general knowledge. In either case, it is the successful location of such paths in memory that will lead the reader to feel, subjectively, that he has "understood" that part of the text.

Moreover, it is only by finding such established or at least intelligible relations between the concepts a text associates that the reader will be able to disambiguate the multiple meanings and syntactic ambiguities of that text, i.e., will be able to recognize exactly what concepts the text is talking about, and what it is attempting to say about them. Finding such paths enables the reader to correct misprints, overlook literal inaccuracies, ignore ridiculous parsings, and carry out all the rest of the processing that he does continuously, and unconsciously, during reading. Some ways that finding such connections in memory can disambiguate words of text will be explored in this and the next chapter, while Chapter VII will discuss cases where finding a path is necessary to determine the structure of the new plane which is to be formed, i.e., to determine how the text should be parsed at that point.

The kind of path locating done by the program of Chapter III is among the freest kind of association of concepts, in that almost no restriction at all is imposed on paths the program is allowed to find between two given concepts. A text, on the other hand, will generally impose restrictions on the paths to

be found between concepts it associates, and more will be said about such restrictions below. For the moment, however, the fact that more restricted paths must often be located need not affect the basic point here: that in order to figure out how to build a new plane to constitute his understanding of what a text is saying, it is in general crucial for a reader to find, already within his memory, some kind of paths connecting all the various concepts the text interrelates.

We emphasize that this is not to say that a reader can only understand things he has already read before, for, if the shortest acceptable connection that he can find in his memory between two concepts is indeed a long "plane-hopping" path (see Section C of Chapter III) this still means that he has found some intelligible connection between those concepts, albeit a connection which, as such, he had not stored or perceived before. (If he is completely unable to find any acceptable connection in memory between concepts we assume he will be unable to understand the section of text which asserts their association. Such failures may occur frequently when the memory search is conducted under great time pressure, as when the reader is skimming, or when the connections between concepts are very obscure, as in some modern poetry. But in ordinary reading a failure to find some connection between concepts should be a relatively rare thing, because of the very high overall interconnectedness of his memory.)

B. THE GENERAL SEMANTIC CONTEXT OF A TEXT

In addition to locating paths between two or more particular words taken as patriarchs, these word concepts need to be related to the larger cognitive content established by all of the prior text. Under the assumption that the reader's memory is indeed utilized as is the memory model in our current

program, such "context" is explainable as follows: As the subject has been reading a text, he has been searching for connecting paths, and hence has been firing activation "spheres" (see Chapter III) from "patriarch" nodes corresponding to many of the words of the text. The activation tags applied in this manner are not all immediately erased, so that they accumulate throughout much of the memory. Suppose that the text is about baseball, so that such tags have accumulated on nodes such as "batter," "ball," "pitcher," and so on. Now, upon encountering, say, the isolated word "strike," the reader fires one activation sphere from the type node heading its baseball meaning, another from the one heading its labor union meaning. Clearly, paths from one of these meanings to intersections with the prior context would pile up more quickly, and tend to be shorter, than those from the other meaning to that context. By finding this best connection, this particular strike is linked to prior context, and, as a vital by-product, the ambiguity of "strike" is resolved almost instantaneously; the reader would say that one of its meanings is "in context," the other not.

Actually it is not necessary to assume in such a theory of association and disambiguation that the reader always fires bursts of activation from the various alternate meanings of a newly encountered word like "strike." For, if the type nodes placed at the heads of alternate meanings of the word, "strike2," "strike3," etc., all had links leading back to a single inclusive node for that word, a path of activation from words read earlier could already have proceeded right to that inclusive node. Then, when the word was recognized in text, it would only be necessary to follow back along the path of activation that had already come into it to find the meaning most consonant with the overall context and its connection to

that context. (It will be recalled that an activation tag is in part a reverse link, so that the processer can always trace back along it). In cases where this sort of linkage occurred the role of the new word would be not so much to introduce a new concept as to select some part of the already established conceptual context to be talked about further. This latter type of disambiguation seems to accord well with our intuitive feeling that in reading (especially reading about familiar subject matter), we do not have to consider the alternate meanings of each word, but just "automatically know" the correct one.[1]

During reading the locating of paths to and within prior memory, and the concomitant disambiguation of words, perhaps proceeds in this latter manner whenever possible, and resorts to the slower procedure of firing bursts of activation through the various meanings of a new word only when that word has not previously been reached by activation. There are some cases in which it would appear that the path to an intersection between prior context and a new word is much shorter if bursts are fired from both.

There are several ways in which the requirement of this section — that a word of text intersect with the text's overall context — can be integrated with the requirement of Section A — that pairs of words in the text intersect with each other. One of these ways will be tested in the next chapter, for the moment we return to the question of how pairs of words are selected for intersection by a text.

---

[1]I am indebted to Dr. Allen Newell for bringing this to my attention.

C. PROVIDING PATRIARCHS FOR PATH FINDING ROUTINES

The theory of text understanding elaborated thus far has ignored syntax.[1] The formulations of Katz and Fodor (1963) imply that the role of syntax is to direct the sequence in which the various words and larger constituents (in the deep structure) of a sentence are to be taken as pairs for semantic processing. The interpretation of this proposal, if a semantic memory is assumed, is that syntactic considerations direct how words are to be chosen as patriarchs for intersecting within the semantic memory. Chapter VI will show that there is confirmation for this general point in the protocol, although "deep structure," in the strict transformational sense, need not be assumed.

The transformationalist theory of "semantic interpretation" further states that, once two words have been chosen for pairwise semantic processing, the particular syntactic relationship between them (such as subject-to-verb or verb-to-object) will also affect the way in which that semantic processing proceeds. The interpretation of this proposal, within a theory assuming a semantic memory, is that syntactic considerations will often direct that, instead of taking some word of the text as a patriarch, some "clue" word (or set of "clue" words) from its defining plane be taken as a patriarch or set of patriarchs. (It will be recalled that clue words are attached to an S, D,

---

[1]Hence it cannot, for example, explain why a reader who encounters, in the middle of a text about baseball, the sentence, "the players' spokesman called a strike," is likely to interpret this as a labor union kind of strike. We will hereafter refer to the process of looking for the first intersections between two words taken as patriarchs (or two sets of words so taken) simply as "intersecting those two words or two sets.

or M parameter symbol to state what is likely to fill that parameter, and hence correspond roughly to the "selectional restrictions" of transformational theory. See Part C, Chapter II).

For example, suppose two words thought to be a verb and its noun object have been selected to be semantically processed as a pair. In this case the noun will be taken as one patriarch. But, the verb itself will not be taken as the other, instead, only the clue words associated with the D parameter symbol in its defining plane will be so taken. This entire set of clue words will then be intersected with the noun, taken as their single opposing patriarch. If there were several possible verb meanings of the verb, then each such meaning may have separate D symbols, each of which has a different set of clue words associated with it. In this case, all these clue words will be taken to form one set of patriarchs, and the intersections to be found between the noun concept and that of one of the clue words will provide basis for disambiguating the verb. In other words, the close connection to some clue word indicates that the particular verb meaning that supplied that clue word is the meaning to select for this sentence.

Tracing an example through in detail will make clear how this sort of mechanism will have to function in a program able to understand text, and also illustrate how our use of clue words differs from the linguist's conception of "selectional restrictions." (See Chomsky, 1965 and Lakoff, 1965, for recent discussions of such restrictions.)

Consider Figure 3. This figure shows two of the meanings for the phrase "called a strike," and the corresponding S

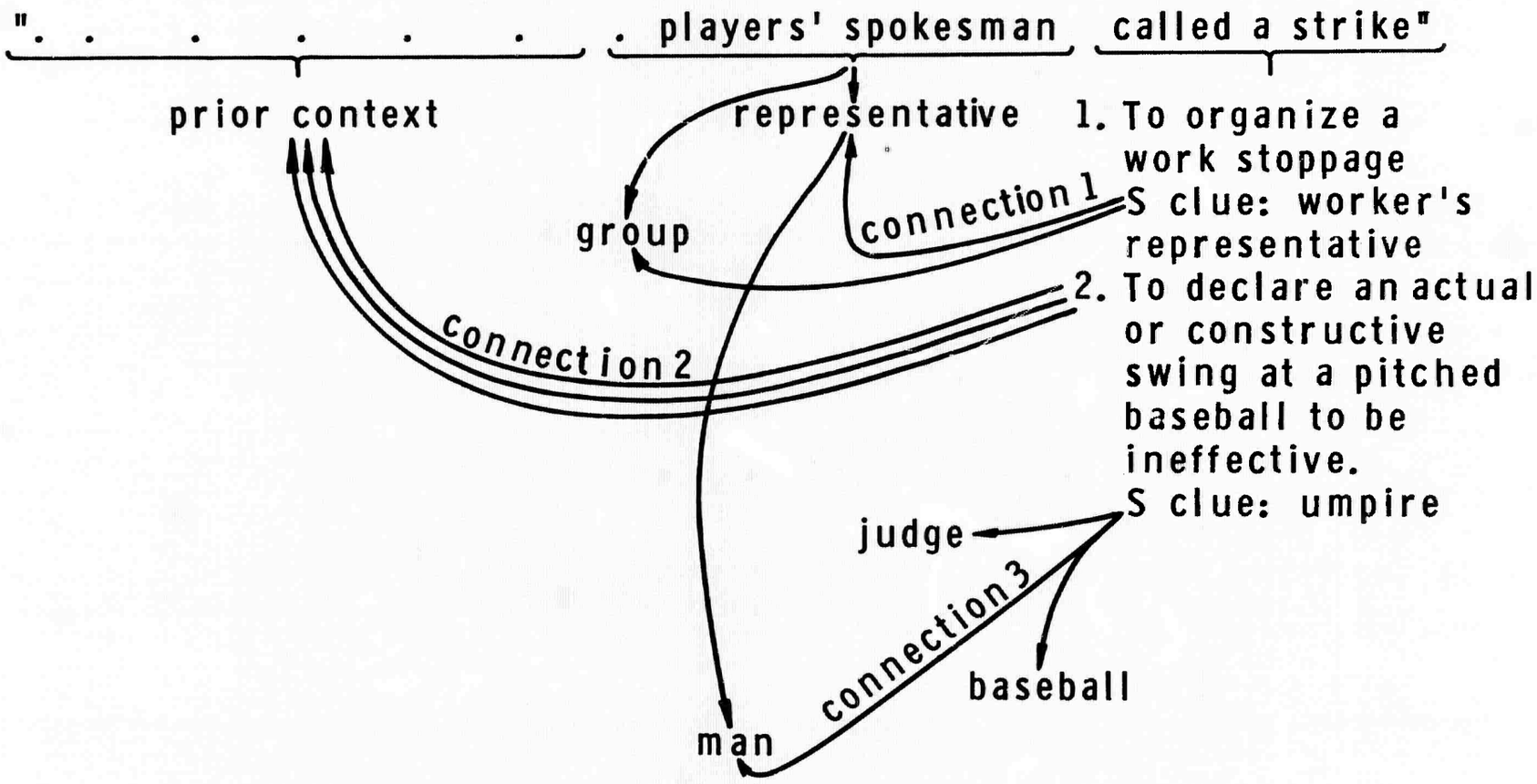


FIG. 3 USE OF PATHS TO DISAMBIGUATE TEXT

clue words that might be associated with each. Suppose it is known from syntactic considerations that the phrase "player's spokesman," is the subject of that phrase. In terms of the theory, this means that "player's spokesman" is to be taken as one patriarch, while the set composed of the two S clues, "umpire," and "workers' representative," is to be taken as the other.

In this case the theory states that, with a correctly formulated semantic memory, some set of paths like that illustrated as connection 1 would be discovered to form a tighter connection than that shown as connection 3, indicating that meaning 1 of the verb phrase offers the most obvious way to render this part of the text intelligible. Indeed, this is the interpretation we would almost always want of such text.

However, just suppose that the "prior context" in Figure 3 were such as to provide the following stretch of text: "John swung hard at the pitch, and was certain his bat had tipped it, but the player's spokesman called a strike." In this case, most readers will want to interpret "called a strike," by meaning 2 rather than by meaning 1 of Figure 3. In terms of our model, this is to be explained as follows: The same procedure as above is followed, up to the point at which connection 1 is located. Then, the conjunction "but" further requires that the entire second clause intersect closely with the preceding clauses of the sentence. By far the strongest connection between these is one like that shown as connection 2, which dictates the opposite interpretation of the meaning of "called a strike." However, this makes connection 1 unacceptable, so a new interpretation of the relationship between "player's spokesman" and "called the strike," must be found. To produce this, the search for intersections between

these two phrases is simply continued (if necessary), until connection 3 is discovered. This path is "thin," but acceptable in terms of connection 2, so that the sentence may be considered interpreted acceptably when it is utilized. The nature of connection 3 would be something to the effect that a "player's spokesman" is among other things a man, and that an "umpire" is also a man.

From this example, it is clear that this theory of understanding places great reliance upon the fact that, in a memory model like that of Chapter 2, intersecting some word taken as one patriarch, with a set of words taken as the opposing patriarch, provides a measure of the relative semantic similarity between the one word and each of the others. That is, the single word's full concept will have intersections to the full concepts of each of the others after differing amounts of search, with shorter paths being discovered first. The only further assumption needed is that a useful measure of "semantic similarity" can be obtained from the length and number of paths connecting two patriarchs. This appears to be generally the case in experiments so far (see Section C of the next chapter) but must remain an hypothesis until a really large semantic memory is set up and tested. Our own judgement is that achieving this sort of similarity judgement will require not only a much larger but also a better structured memory, as well as a more discriminating search routine, than our program has used so far (see Chapter VIII and Appendix III). But, given a few more tries, such a measure would certainly not appear to be beyond reach. The reason that in the memory model all information is stated homogeneously in terms of word concepts, is precisely so that "clue words" in one concept can always be taken as patriarchs and intersected with other words of the text. Then, the paths to an intersection that are found

provide basis for a decision about how that part of the text is to be interpreted.

One point to note here is that this produces a considerably different picture of "understanding" from that implied by "selectional restrictions," when these are stated, as in current linguistics, simply as properties like "animate," "abstract," etc.[1] Rather than restricting what some word's, say, object can be, clue information states only what its object is most likely to be. Given any sentence, such a memory thus provides a way of selecting the simplest interpretation of that sentence available, but nevertheless can also interpret the sentence in other ways, by means of long paths, if short ones are not available, or if the interpretation by shorter paths conflicts with other connections, as in the example above.

Therefore, so-called "anomalous" phrases and sentences, such as "silent paint," become simply those that require relatively long, weak paths to interpret intelligibly. Such interpretation requires no mechanism different from that employed in the interpretation of non-anomalous sentences. We submit that people in fact operate this way, rather than, as the work of Katz and his co-authors would imply, by rules that declare every sentence containing some unorthodox combination

---

[1] In some cases, apparently all that can be stated as a clue word is some very general notion such as, say, that the value of some S parameter is likely to be a person. In such a case the clues are much like stating simply that the subject of this word must be "human," except that the memory store will still select a human subject first, a non-human but animate subject next, and then a non-animate subject at still a weaker path.

of concepts to be "anomalous," leaving those so declared to be interpreted by a totally separate set of processes, which would somehow interpret all such sentences by "analogy." (Chomsky, 1965, page 149.)

In such a performance model decisions about word meanings based on path-lengths would always be heuristic, not algorithmic, in that they would merely be selections of some particular interpretation as more probable than others, given the other contextual information considered, and the total information stored in the memory at the time of reading.

However, the question still remains as to just how syntactic considerations select particular words of a sentence, and types of clue words within those words, to be taken as patriarchs for intersecting in particular cases. On this question, one would expect to get the most help from modern linguistics, but in fact gets very little. The reason is that generative grammars, although serving to describe sentences of a language, tell nothing about how people may obtain such a description, given a sentence. (In Section B of Chapter VIII one proposal made in this regard, that of Miller and Chomsky, 1963, is considered, and reasons why it is not acceptable are given.) The various attempts to build an automatic parser (See Kuno, 1965, Bobrow, 1963, and Hays, 1966) may eventually provide help on this issue, but they do not yet seem to. This is primarily because, lacking any semantic theory at all, these parsing programs have had to assume that no semantic processing whatsoever is employed during syntactic processing. The main concern of Chapter VII will be to find within the protocol some empirical indication of how syntactic facts direct the semantic intersection process.

In this chapter, then, a certain theoretical view of what it means to understand text has been presented, building primarily on the path finding process of Chapter III. This view asserts that in general understanding is equivalent to construction, in the brain of the reader, of new "planes of tokens" like those employed in the memory model. The plane building process is guided by discovering, within the general semantic memory that the reader must already have, acceptable paths between concepts the text discusses. In Section B one way that running text could be linked to the overall "context" established by prior parts of that text was considered. In Part C the use of clue words associated with the model's "parameter" symbols as patriarchs was considered.

Although we have been able to agree with the transformationalists that syntactic clues serve to guide the combinations in which words of a sentence are semantically processed, one cannot, in a performance model, simply assume that there will always be a convenient deep structure at hand to do this directing. One might assume that someone, someday will succeed in building a parser to provide such deep structures, although we do not think this is a reasonable hope (see Chapter VIII), and instead will consider as an open question how syntactic facts about a given sentence help direct semantic interrelating of its words. Thus, a major problem that this chapter's theoretical view poses for our empirical analysis of the protocol is: how do syntactic properties of words and sentences feed the right combinations of patriarchs to path

finding routines. After further preliminary analyses in the next chapter, this problem is returned to in Chapter VII.[1]

---

[1]Given the great flexibility with which any given assertion can be expressed in a natural language, any really human-like model of text understanding must also be very good at recognizing complete or partial synonymy (and contradiction) between a statement in memory and one in text. Doing this is also possible within the sort of memory being considered here, although it will not be discussed in detail in this thesis. (See Appendix III on the essentials of how synonymy may be recognized by use of the model.)

CHAPTER VI

PRELIMINARY ANALYSIS OF THE PROTOCOL AND OF PARSING AS A PROBLEM SOLVING TASK[1]

A. DIVIDING THE PROTOCOL INTO EPISODES

The first step taken in analysis of AX s protocol was simply reading it as thoroughly as possible, attempting to characterize what she seemed to be doing at each numbered paragraph. After going through the protocol many times in this way, it began to appear that AX repeats certain processing steps at various points. By giving names to these repeated steps a set of process categories was evolved, which have stabilized into the set of 13 described in Part D of Appendix 1. In the right column of Part C these categories have been used to characterize each step AX appears to take in the protocol. The categories are merely descriptive, i.e., are condensed

---

[1]The general orientation toward protocol analysis, as well as many of the specific techniques used in this chapter, derive directly from techniques worked out by Newell and Simon. See especially Newell and Simon (1964).

restatements of what AX is doing at each particular point.[1] Such descriptive categorization serves to reduce the initial complexity of what AX is doing during the protocol to a manageable number of processes, and allows recurrences to be noted.

For our analysis the most important category is the one called "Seg." This category was created to account for the fact that AX, in encoding the text given to her, persistently reads aloud short segments of that text. There is no necessary reason for this in the encoding procedure, where words are dealt with one at a time. In fact, for studying the understanding process, it is one advantage of the encoding notation that it forces a coder to deal at some stage with each word's relationship to other words one at a time. Since part of the instructions given to AX suggested that she first read aloud each definition, no significance can be attached to her reading whole definitions, and these are not characterized as "Seg." Nevertheless, one notes in the protocol that AX very persistently reads aloud not just single words and complete definitions, but two to six word segments of text.

After she "bites off" a segment in this way, the protocol typically shows AX proceeding to encode on the plane the link which connects the new segment to the previously formed

---

[1]The only category which evolves an inference in being applied is "Cf," it being inferred that AX has compared a parameter's clues to a potential value for that parameter before filling the parameter with it. This is done to point up the protocol's relationship to the theoretical orientation described in the preceding chapter; "Cf" could be replaced with a purely descriptive category with no essential change in the following analyses.

structure, and then the other links that connect the words of the new segment together. After this is completed she proceeds to "bite off" another segment of text and encode it similarly.

Thus, the encoding of segments defines natural episodes within the protocol, each of which begins with a "Seg." step. In order to see these episodes and the segments they pertain to clearly, they have been numbered (S1, S2, and so on) and the segment itself quoted in the right hand column of the protocol.[1] These segments are shown in Table III, designed to show clearly which pieces of the running text AX bites off for consideration. These segments accord well with most standard linguistic formulations of syntactic constituents, forming prepositional phrases and predicates rather than subject-verb or subject-preposition units. For example, there is not a single case in which AX groups a noun with a modifying preposition while omitting the preposition's object. Thus, AX's segments would appear to confirm the general proposition of Katz, et al that semantic processing operates on syntactically related groupings. However, to be able to name AX's segments is still far short of having a set of rigorous rules which,

---

[1]On a few occasions AX reads a long string of text, but then immediately, before encoding any of it, delimits a smaller subsegment of this string to process; in this case only the smaller segment has been characterized as "Seg." (see #7, #8 and #18, #19). Also, AX sometimes reads a stretch of text after she has encoded it, in which case she is clearly testing the English sense of what she has encoded. In this case the step is characterized as "Test" rather than "Seg." Only continuous segments of the text, which AX herself, during editing of the protocol, spontaneously decided to enclose in quotes, have been considered eligible to be characterized as "Seg." Once a segment has once been characterized as "Seg," later occurences of it in quotes are not characterized again as "Seg."

TABLE III

THE TEXT AS SEGMENTED BY AX

(Nil indicates that the text was not contained in any segment)

Whip:

Nil Stick
S1 with cord
S2 cord or leather
S3 fixed to end of it
S4 fixed to end
S5 end of it
S6 used to give blows
S7 give blows in driving animals
S8 give blows
S9 in driving animals
S10 driving animals
S11 animals etc.

S12 as punishment.
Nil Give blows to
S13 with whip.
Nil Be
S14 acting as whip
S15 as whip
S16 to(dogs,political group).
S17 dogs,political group

TABLE III (Cont'd)

whip 3:

```
S18  Person responsible
S19         responsible for seeing that others of his political group
S20                                      others of his political group
S21                                            of his political group
S22                                                   political group
S23                                                                  in Parliament
S24                                                                                  are present
S25                                                                                      present
     when desired,
S26  when desired,
S27                 do the right thing.
Nil                                    Note
S28                                         from whip
S29                                                  requesting person to be present
S30                                                               person to be present
S31                                                                    be present in Parliament
S32                                                                       present in Parliament
S33                                                                               in Parliament
S34                                                                                              etc.
```

whip 7:

```
S35  Give (eggs etc.) quick blows
S36             etc. (= "or food")
S37                                with fork etc
S38                                     fork etc
S39                                          etc(= "or spoon")
S40                                                get them mixed with air
S41                                             to get them mixed
S42                                                       mixed with air
S43                                                             with air
S44                                                                      so that they become stiff
S45                                                                      so that
S46                                                                              they become stiff
```

TABLE III (Cont'd)

whip 9:

S47 go or take quickly (out, through)
S48 out, through
S49 out, through etc.

given the same text, would segment it as AX does. Such rules are what are required in an explicit theory or a simulation program, and the next chapter will consider such rules, after the problem that faces a reader as he tries to comprehend text has been set up more explicitly.[1]

B. THE COMPREHENSION SPACE OF A SENTENCE

In the memory model, every word's full concept contains information showing its various possible meanings, and clearly a human subject is also able to recognize separate meanings for each word. Table IV and Figure 4 have been created as analytical tools to enumerate the range of choices such mental information would provide for a coder or a reader. That is, in order to make explicit the assumption that the coder's memory provides her with choices similar to those in a dictionary, the investigator looked up (in a Basic English Dictionary) each word that appeared in the definitional text AX had been given to encode (excluding prepositions and articles). Then all of each word's meanings (that were compatible with the inflections present on the word) were

---

[1]The way AX segments the text is also related to our assumption that the processor she uses to understand text is not itself consciously accessible to her, that only its results are. She seems to find it necessary to hand that processor a segment of text, and then look at its results to decide how to construct the plane. (In paragraphs' such as #158, e.g., it seems apparent that she is feeding different segments to such a processor, getting results which "make sense" or do not, and then coding on the basis of these.) The accessible outputs of her inaccessible processor seem each to be a sort of transitory mental representation, in our terms, a part of a plane of token nodes. We assume that one reason AX divides the text into segments is to be able to retain these mental representations clearly enough to write down their details accurately.

TABLE IV

POSSIBLE MEANINGS AND PARAMETERS FOR
EACH WORD OF THE TEXT AX ENCODED

stick
  stick1 (to jab)
    S person, pointed object
    D person
    M
  stick2 (any rod-like object)
    M
  stick4 (to place quickly)
    S person
    D small possession
    M
  stick5 (to (cause to) adhere)
    S paper, person
    D paper, wood
    M

with
    M
    D

cord
  cord1 (twine, thong)
    M
  cord2 (to tie with cord)
    S person
    D package
    M

or
    M
    D

leather
    M

fixed
  fix(1)ed (held steady)
    S person
    D) object, color, picture
    M)
  fix(2)ed (attended to)
    S person
    D attention, eyes
    M
  fix(3)ed (repaired)
    S person, solution
    D trouble, broken object
    M

to
    M
    D

end
  end1 (terminating point)
    M point
  end2 (to (cause to) terminate)
    S person, event
    D event
    M

of
    M
    D

it----------Find a referent for this word

,
  ,1 (or)
    M
    D
  ,2 (and)
    M
    D

used
    S person, engine, process
    D) instrument
    M)

```
to
    M
    D


give
  give1          (provide with)
    S person, document
    D object, emotion
    M
  give2          (make, as in "give a jump")
    S person
    D action
    M
  give3          (elasticity)
    M
  give4          (to stretch or bend)
    S flexible object
    M


blows
  blow(1)s       (moves by using air)
    S wind, person
    D object, nose
    M
  blow(8)s       (sudden impacts)
    M
  blow(9)s       (windstorms)
    M


in
    M
    D


driving
  driv(1)ing     (forcing or directing)
    S person, power
    D person, vehicle, machine
    M
  driv(2)ing     (hard working)
    M person


animals
    M
```

etc----------Assume an "or" preceding this word, replace this word with a concrete term.

or
M
D

as
M
D

punishment
M

Give
give1 (provide with)
S person, document
D object, emotion
M
give2 (make, as in "give a jump")
S person
D action
M
give3 (elasticity)
M
give4 (to stretch or bend)
S flexible object
M

blows
blow(1)s (moves by using air)
S wind, person
D object, nose
M
blow(8)s (sudden impacts)
M
blow(9)s (windstorms)
M

to
M
D

with
    M
    D

whip----------Substitute head node of the definition.

Be
  bel (to have as property or name)
    S
    D quality, position, relationship, name
  be2 (to exist)
    S
    M

acting
  act(1)ing (effecting, affecting)
    S person
    D behavior, process
    M
  act(2)ing (substitute)
    M authority
  act(3)ing (pretending)
    S person
    D role
    M

as
    M
    D

whip----------substitute head node of the definition.

to
    M
    D

```
dogs
  dog(1)s          (the four legged animals)
    M
  dog(2)s          (trails closely)
    S animal
    D person
    M


,
  ,1               (or)
    M
    D
  ,2               (and)
    M
    D

political
    M organization, person, act


group
  group1           (an aggregate)
    M norm
  group2           (to form into a group)
    S person, theory
    D objects, people
    M


Person
    M


responsible
    M


for
    M
    D


seeing
  see(1)ing        (sighting)
    S person
    D object
    M
```

see(2)ing (comprehending)
S person
D idea, significance
M
see(3)ing (able to see)
M
see(4)ing (visionary)
M
see(5)ing (meeting with)
S person
D person
M
see(6)ing (insuring)
S person
D event
M

that
that1 (conjunction)-----Recurse, i.e., comprehend the following clause and treat it as a unit.
that2 (article)
M
that3 (pronoun)---------Substitute referent.
that5 (as)
M
D

others---------Substitute referent.

of
M
D

his----------Substitute referent, assume "of" before the referent.

political
M organization, person, act

group
  group1 (an aggregate)
    M norm
  group2 (to form into a group)
    S person, theory
    D objects, people
    M

in
    M
    D

Parliament
    M

are
  are1 (to have as property or name)
    S plural
    D quality, position, relationship, name
  are2 (to exist)
    S plural
    M

present
  present1 (here, now, there)
    M
  present2 (to put forward)
    S person
    D gift, person, idea
    M
  present3 (thing given)
    M

when
  when1 (at a time)
    M event
    D event
  when2 (although)
    M event
    D situation or event contrary to M

```
desired
    S person
    D
    M


,
  ,1              (or)
    M
    D
  ,2              (and)
    M
    D


do
  do1             (perform)
    S person
    D act, work
    M
  do2             (be adequate)
    S object, act
    M


the
    M


right
  right1          (correct)
    M
  right2          (to make correct)
    S person
    D situation
    M
  right3          (opposite left)
    M


thing
    M


Note
  note1           (a short record)
    M
```

note2 (to observe or record)
S person
D fact
M
note3 (musical)
M
note5 (quality of voice)
M
note10 (paper money)
M

from
M
D

whip----------Substitute referent.

requesting
M)
S) person, agency
D event, object

person
M

to
M
D

be
be1 (to have as property or name)
S
D quality, position, relationship, name
be2 (to exist)
S
M

present
present1 (here, now, there)
M

present2 (to put forward)
S person
D gift, person, idea
M
present3 (thing given)
M

in
M
D

Parliament
M

etc.----------Assume an "or" preceding this word, replace this word with a concrete term.

Give
give1 (provide with)
S person, document
D object, emotion
M
give2 (make, as in "give a jump")
S person
D action
M
give3 (elasticity)
M
give4 (to stretch or bend)
S flexible object
M

eggs
egg(1)s (as from a chicken)
M
egg(2)s (taunts, encourages)
S person
D persor
M

etc.----------Assume an "or" preceding this word, replace this word with a concrete term.

```
quick
    M act, animal


blows
  blow(1)s      (moves by using air)
    S wind, person
    D object, nose
    M
  blow(8)s      (sudden impacts)
    M
  blow(9)s      (windstorms)
    M


with
    M
    D


fork
  fork1         (to branch)
    S road
    M
  fork2         (the eating implement)
    M
  fork3         (to spear with a fork)
    S person
    D hay, food
    M


etc.----------Assume an "or" preceding this word, replace
               this word with a concrete term.


to
    M
    D


get
  get1          (to acquire, become or understand)
    S person
    D condition, object, point
    M
  get2          (to cause)
    S person
    D change
    M
```

them----------Substitute referent.

mixed
S person, process
D, M } substances, objects

with
M
D

air
air1 (the gas we breathe)
M
air2 (to expose to air)
S person
D clothing, room
M
air3 (an appearance)
M

so
M
D

that
that1 (conjunction)----------Recurse, i.e., comprehend the following clause and treat it as a unit.
that2 (article)
M
that3, that5 (pronoun, as)----------Substitute referent.
M
D

they----------Substitute referent.

become
become1 (get to be)
S situation, person
D condition
M

become2 (are flattering to)
S clothes
D person
M

stiff
stiff1 (rigid or formal)
M
stiff3 (strong)
M alcoholic drink

Go
go1 (to move, operate)
S person, vehicle, machine
D place
M
go4 (to extend)
S road
D place
M
go5 (become, develop)
S
D condition
M
go6 (be used up, break)
S resource, implement
M
go7 (be normally kept at)
S object
D place
M

or
M
D

take
take1 (to acquire)
S person
D
M
take2 (to accompany in order to move)
S person
D
M

take8 (have desired effect)
S innoculation
M
take10 (react to in certain way)
S person
D news, suggestion
M
take13 (to photograph)
S person, camera
D picture, scene
M
take17 (require)
S task
D material, time, knowledge
M

quickly
M action

out
out1 (outside of or in public)
M
out8 (not current)
M old object, procedure

,
,1 (or)
M
D
,2 (and)
M
D

through
through1 (between)
S
D two things
M
through3 (with aid of)
S event, achievement
D person
M
through4 (finished)
M event

etc.----------Assume an "or" preceding this word, replace this word with a concrete term.

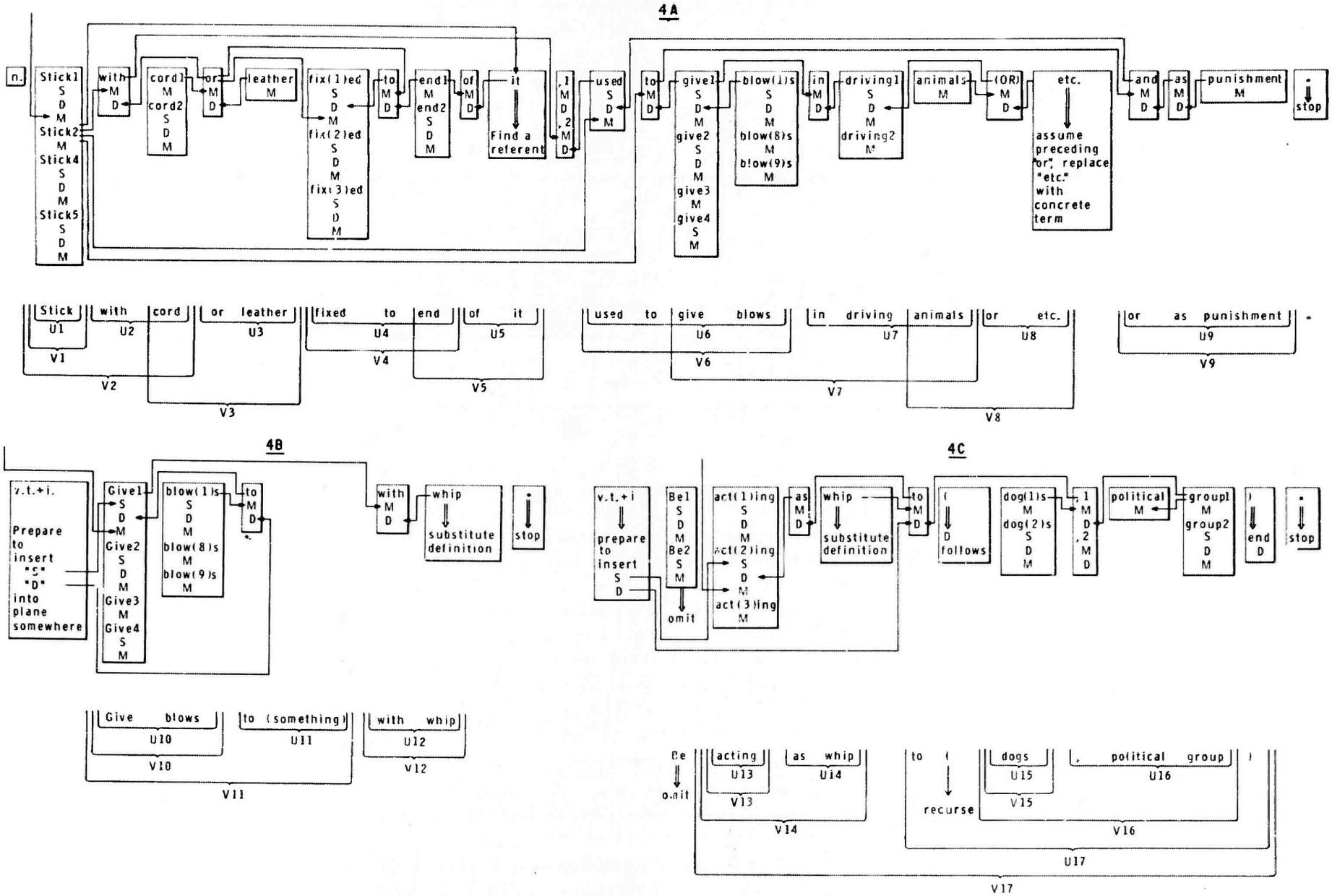


FIG.4 ENCODING OF THE TEXT BY SUBJECT AX

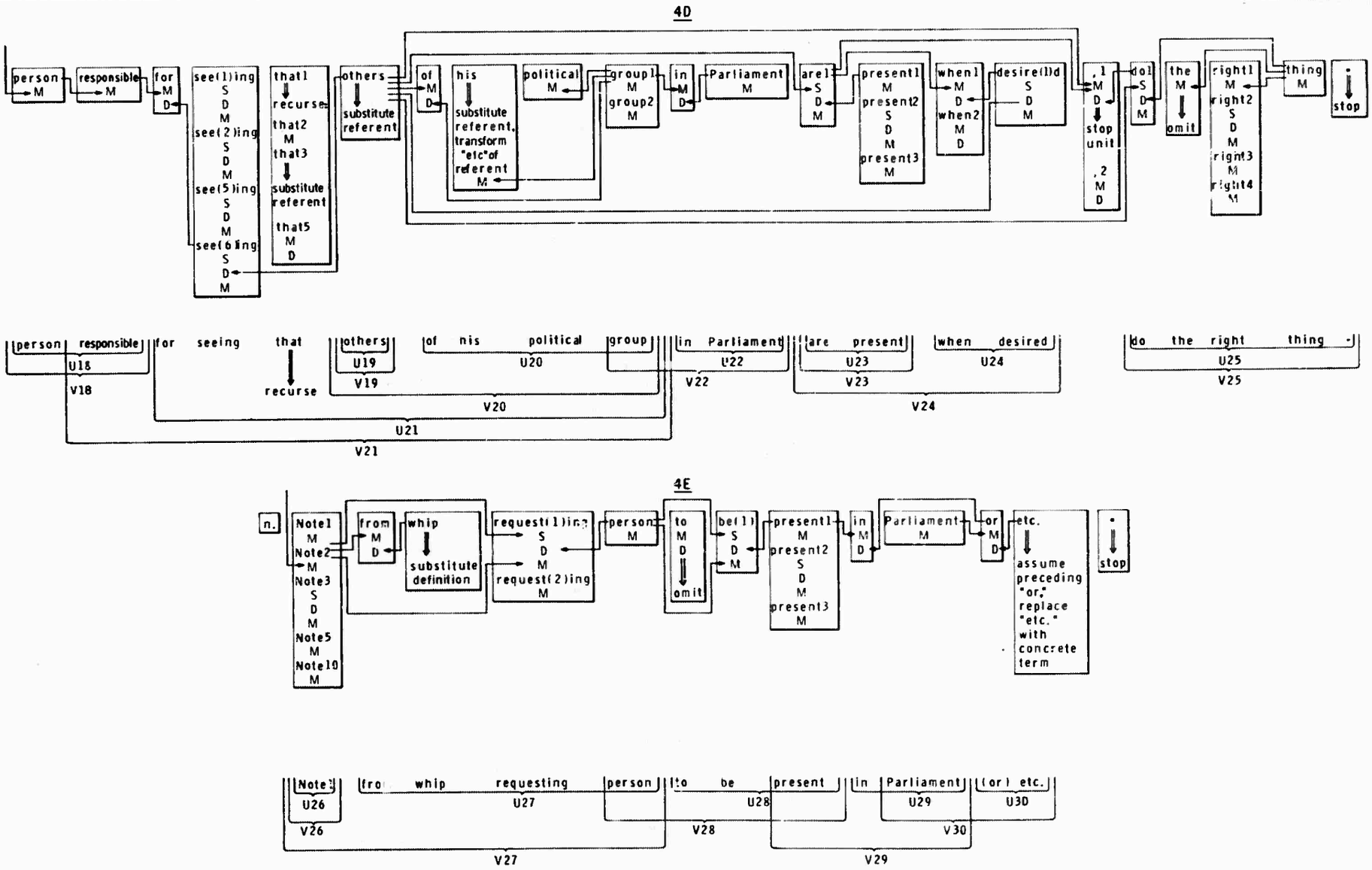


FIG.4 ENCODING OF THE TEXT BY SUBJECT AX

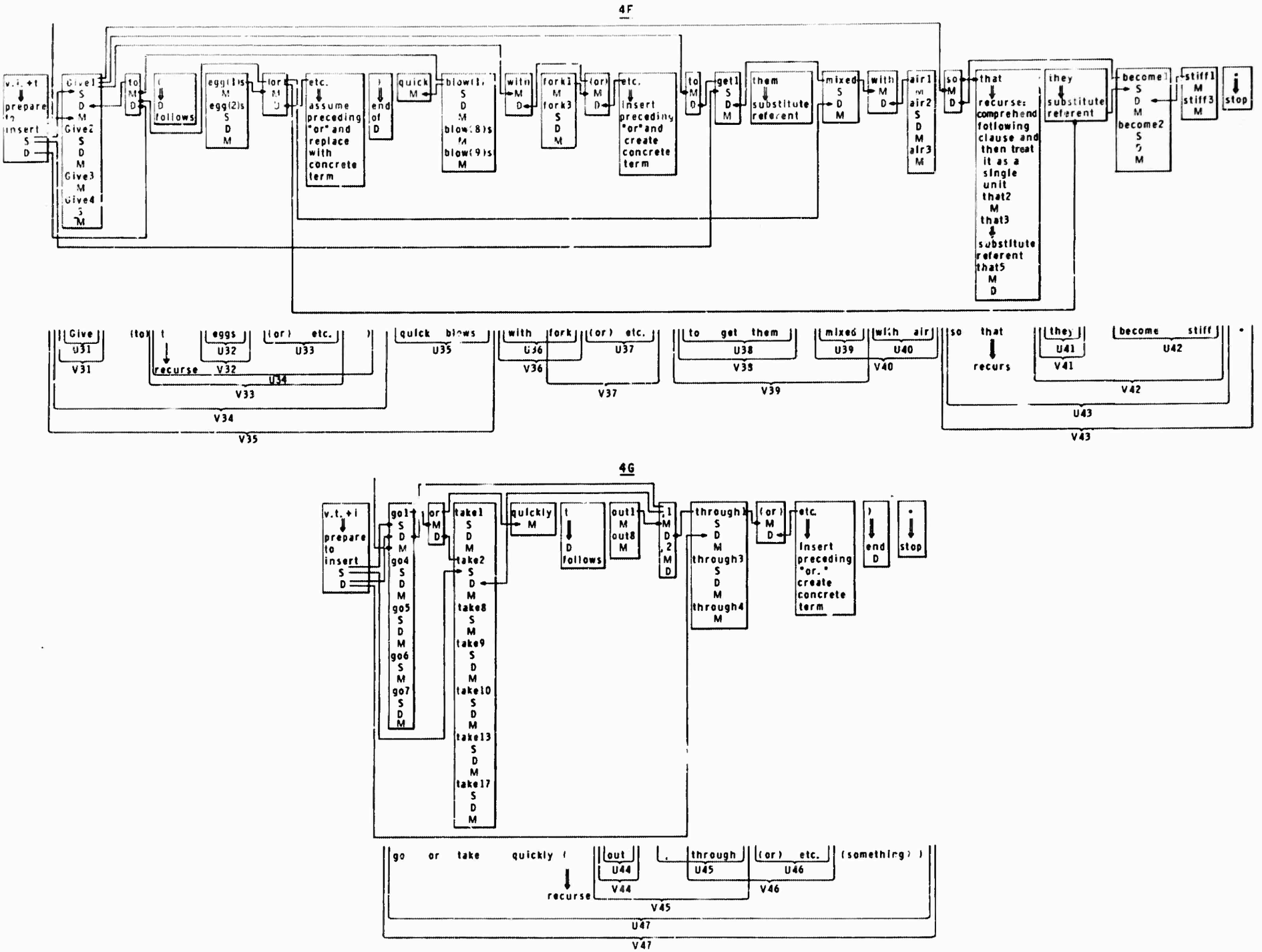


FIG. 4 ENCODING OF THE TEXT BY SUBJECT AX

listed.[1] (AX, of course, never saw this information; all she was given to work with was the text, Part A of Appendix I. The reason for not considering prepositions polysemantic at this stage will be discussed in Chapter VII.)

After listing each word's meaning, the next step in creating Table III was to list, for each sense of each word, the S, D, or M parameter symbols that its full concept would apparently have to contain somewhere. (It will be recalled that an M parameter represents not a modifier of the word containing it, but something the containing word modifies. Thus, if word A is selected as the value of an M parameter in word B, this means that B modifies A.)

The final step in creating Table III was to attempt to list, for many of these parameters, a "clue" word or two stating what seems to be a likely sort of thing to fill that parameter. Often these clue words were included in the dictionary definition of the word, in other cases we simply added them from our general knowledge. If no such clue concept came readily to mind it was omitted; their purpose here is only

---

[1]Actually, a judgement was made at this point as to how many of the word's separate dictionary senses seemed significantly distinct to merit being listed separately. In these judgements, the same criteria were applied as are customarily used in deciding which meanings to keep separate for a word which is to be encoded (see Chapter IV), with one exception. This is that for Table III, whenever such dictionary meanings of a word differed in parts of speech (other than by an adjective-noun distinction) these senses were listed separately. This change in the usual criteria was made because Table III is designed to enumerate explicitly the separate possibilities available to a coder or reader as he understands text. In Table III the brief statements after each numbered sense indicate what that particular meaning is.

illustrative. However, one should remember in looking at one of these clue words that its function is not to limit what can be, say, the subject of some word, but rather to give some basis for selection of one alternative out of a set of candidates when these are provided by a text in which the word is used (see Chapter V).

Figure 4 was then formed by laying out the information of Table III in normal left-to-right order and adding punctuation. The long solid arrows in Figure 4 show the actual "parameter fillings" (word-to-word relationships decided upon by AX during encoding).[1]

Besides its use to show how AX understood the text, Figure 4 allows a method of enumerating all the possible ways of comprehending these sentences, in the same way that all the possible moves in chess or in solving well defined problems can be enumerated. That is, with Figure 4 one can calculate the number of conceivable relationships that could be formed between the words (as inflected) of these sentences of text. Then, various combinations of these word-to-word relationships constitute conceivable "comprehensions" of the sentence. The total set of such combinations of relationships within a sentence will be called its "comprehension space," since to actually understand the sentence is to utilize some one (or possibly two) of these combinations from this set of possibilities. Actual comprehension is done by using word order,

---

[1]In Figure 4 a downward pointing arrow within the box enclosing a word or punctuation mark means that we assume that that word or punctuation functions as a signal to a person's understanding processor that the action described below the arrow should be performed. Green (1961) and several more recent programs have advantageously defined certain words as cues for some kind of action.

inflectional interdependencies and semantic interrelationships to rule out many comprehensions while selecting one or two; to enumerate the total comprehension space of a sentence is, therefore, to count the combinations of word-to-word relationships possible on the basis of its inflected words alone, without taking any syntactic or semantic interdependencies into account.

To carry out a calculation of the comprehension space of the first sentence in Figure 4, we observe that "stick," the first word, provides ten parameter symbols: 3 for "stick1," 1 for "stick2," 3 for "stick4," and 3 for "stick5."[1] The total number of ways "stick" can be related to other words of the sentence is the total number of ways its parameters may be filled by taking those words as values. If we assume that the sentence is to be comprehended without any "double meaning," then only one of the four meanings of "stick" will be chosen in any one comprehension. If "stick1" is chosen, three parameter symbols will need to be filled. If "stick2" is chosen, only one parameter symbol will need to be filled, etc. Suppose "stick2" is chosen. Then there will be K ways "stick" may be related to the rest of the sentence, where K is the number of possible values available to fill the single parameter of

---

[1]What one might call the total "parsing space" of a sentence is a subset of its comprehension space, since, for parsing, different meanings of words are ignored, except as these provide different word types, i.e., different kinds of parameters. Thus the parsing space of a sentence shown in Figure 4 can be calculated by the procedure we outline below for calculating its comprehension space, if each particular type of parameter, S, D, or M, is counted only once in a word concept, even though it occurs more than once in different meanings of the word. By such counting, "stick" provides three rather than ten parameters.

"stick2." On the other hand, if "stick1" is chosen, there will be K^3 ways that "stick" may be related to the rest of the sentence, since any permutation of these possible values provides a different way that "stick1" can be related to the rest of the sentence. Since we do not know in advance which meaning of "stick" will be chosen, there are initially $K^3 + K + K^3 + K^3$ ways that its parameters may be filled. In this sentence, before any analysis limits the choice of values to fill the parameters of "stick," there would appear to be 22 possible values for each of its parameters. (This 22 excludes the word "stick" itself, but includes all the other words, and those punctuation marks which can form combination units which in turn can fill a parameter.) The possibility of not filling a parameter must be added to these 22, giving K=23. Thus, it is conceivable, before any syntactic or semantic analysis at all, for "stick" to be related in any one of $23^3 + 23^1 + 23^3 + 23^3 =$ 36,524 ways to the rest of this sentence. This figure excludes the possibility of puns, but allows for the possibility of a word being the value of more than one parameter, which is entirely possible (see Figure 1-a, for example).

However, for our present purposes the exact magnitude of K is unimportant, as well as is whether or not K's size should be different for different words of the sentence, e.g., should decrease as one moves along in the sentence. (For a calculation of the total comprehension space, little if any such change in K should be made, for, again, parameter values do not get "used up;" a single word of a sentence can appear at several places in the parsing structure of the sentence, hence filling several parameters.) The second word of this text, "with," provides two parameters to be filled, and hence could be related in K^2 ways to the rest of the sentence. Similarly, the third word could be related in $K + K^3$ ways, etc.

Now, to get the total number of "comprehensions" of this sentence all the combinations of the relationship possibilities for each word must be calculated. Carrying out this calculation will yield a number whose exact size will depend on K, but which clearly will be astronomical. It seems clear that the space of possible comprehensions for any moderately long sentence will be of similar size. Thus it seems that for sentence comprehension, just as for chess playing and most other interesting problems, there is no real possibility for either a human or a computer to actually enumerate the total space of possible text comprehensions. (Cf. the summary of the heuristic programming literature in Newell and Simon, 1963.)

Even though most of a texts' comprehension space must, therefore, never actually be considered by a human language comprehender, the existence of some very long lines in Figure 4 shows clearly that, in selecting one particular comprehension of a piece of text, a person must have open a very sizeable number of possibilities. The heuristic methods by which one particular comprehension of text is selected is the central problem for anyone who would explain "understanding," just as the heuristic methods by which one particular chess move is selected from all those possible is the central problem for anyone who would explain chess playing. We turn now to possible ways of selecting one such comprehension.

C. THE NO-SYNTAX HYPOTHESIS

One extreme view of how a single comprehension of text may be arrived at is that a purely syntactic parser produces all the syntactically correct parsings of each sentence, before any semantic processing at all is attempted. This sort of assumption is implicit in attempts to develop automatic

parsing programs, and would seem to be implied in the formulations of Katz and his co-authors, if these are extended in any straightforward way to performance models (which transformationists do not advocate.) A good deal of effort has been and still is being spent on developing automatic parsers, with moderate success. (For a survey of these efforts see Bobrow, 1963. See also Hays, 1966.) However, such parsers are as yet by no means wholly successful in parsing ordinary, unselected text, and presumably most of the people working on such parsers now realize that they are attempting a task which, in postponing all consideration of semantics, is almost surely unlike a human's language processer. (What is perhaps the most successful of these parsers has in fact introduced semantics to some degree by proliferating syntactic word types — nouns, verbs, etc. — until the program now employs, instead of the traditional eight "parts of speech," over two hundred word types. (On this program, see Kuno, 1965, and Appendix II.)

In this section an experiment will be described that explores the opposite extreme assumption: that syntax serves almost no initial role at all for the understander of text. This "no-syntax" hypothesis is no less unrealistic than is assuming that language should be processed without any early use of semantic knowledge. But, in order to see exactly what could be accomplished with a semantic memory alone and almost no initial syntactic processing, a large series of intersections were run, using a special version of the memory model and intersection program[1] of Chapter III.

---

[1] The program used in these tests was written by Dr. D. G. Bobrow in LISP 1.5. (See Berkeley and Bobrow, 1964; and McCarthy et al, 1962.)

As input, this program was given the major words of the same text that subject AX encoded — the Basic English Dictionary definitions of "whip." It was also given the order in which these words appeared in sentences of that text. The program's task was to properly disambiguate the words of this input that had more than one meaning. In order to provide the program with a memory model containing the necessary information, each of these words had to be looked up, and its definition(s) encoded (in a very abbreviated way, which omitted most of the structure of a plane, specializing suffixes, etc.) The planes resulting from this were then interlinked appropriately to form the network comprising a memory model. This memory contained definitions of the 35 distinct "context" words (69 distinct definitions) which appear at the top of the columns in Figures 5-a through 5-e. Observe that these columns are arranged in the same order that their words appear in the text given to AX.

The other body of semantic knowledge which is most relevant to understanding this text is some prior, independent knowledge of the various concepts which "whip" can refer to. While this prior knowledge should contain approximately the same information as that in the text AX encoded, it could not be expected, in any real situation, to be stated in a form identical to that of the Basic English definitional text. In order to get something into the memory corresponding to this independent knowledge of "whip's" possible concepts, the word "whip" was looked up in another dictionary, and its definitions there encoded and incorporated into the model memory. These (Webster's) definitions — shown in full in the note to Figure 5 — differ from the text given to AX to encode in that they employ quite different wording, and organize the information as three distinct meanings of whip instead of four.

Figure 5: Computer Located Intersections Between
Words of Running Text and Three Independent Concepts of "Whip"

Note: The words along the top row in these tables come from the Basic English definitions of "whip," while "whip1," "whip2," and "whip3" in the left column refer to definitions abstracted from Webster's New Collegiate Dictionary, second ed., 1959. These read as below. All underlined words also had their definitions included in the memory model.

Whip:

1. To move, take, pull, snatch, jerk or the like, suddenly and forcibly.
   To beat (eggs, cream or the like) into a froth, as with a whisk, fork or the like.
   To move nimbly; to start, turn, go, pass or the like, quickly or suddenly; whisk.
2. An instrument consisting usually of a lash attached to a handle used in whipping
   To strike with a lash, whip, rod or the like, lash, beat.
   To punish by lashing, flog.
   To force, urge or drive by the use of a whip, rod, etc.
3. To gather together or hold together for united action, in the manner of a party whip.
   A person, as a member of a legislature, appointed to enforce party discipline, and to secure the attendance of the members of a party at any important session.

Scoring Key for Intersection Depths

1, 1 = 3
1, 2 = 1

5A

	Stick			Cord	Leather	Fix			End	Use	Give			Blow		Drive		Animal	Punish	TOTALS
	to jab, rod-like object	to place quickly	to adhere			held steady	to attend to	to correct			to provide	to carry out	to stretch, elasticity	to puff, windstorm	Impact	to force, to dr[illegible]	hard-working			
	1#*	2	3			1#*	2	3			1#*	2	3	1?	2#?	1#*	2			
Whip1 to move something suddenly	⑦ 1,2 1,3 1,3 1,2 1,3 1,2 1,3	③ 1,2 1,2 1,2	④ 1,1 1,3 1,3 1,3	④ 1,2	② 1,2 1,2	⑥ 1,2 1,3 1,2	① –	③ 1,2	⑥ 1,2 1,3 1,2 1,3 1,2	⑪ 1,3 1,3 1,3	⑩ 1,3 1,2	① 1,2	⑤ 1,3	⑩ 1,1	⑤ 1,2	⑩ 1,2 1,2 1,3 1,2 1,1 1,1 1,2	⑩ –	① 1,3	② 1,2 1,2	
	3	3	3	1	2	2	0	1	3	0	1	1	0	3	1	10	0	0	2	36
Whip2#* to lash or a lash	⑨ 1,1 1,3 1,2 1,2 1,3 1,2 1,3 1,3 1,2	③ 1,3 1,2 1,3	④ 1,2 1,2 1,3 1,2	⑤ 1,3	② 1,3 1,2	⑧ 1,3 1,2 1,2 1,2	② –	③ 1,2	⑤ 1,3 1,3 1,3 1,3	⑪ 1,2 1,2 1,2	⑩ 1,2 1,2	② 1,2	⑤ 1,2	⑩ 1,3	⑤ 1,3	⑯ 1,2 1,2 1,2 1,2 1,2 1,2 1,1 1,2 1,2	① –	① 1,3	⑥ 1,2 1,2 1,2	
	7	1	3	0	1	3	0	1	0	3	2	1	1	0	0	11	0	0	3	37
Whip3 political party whip	④ 1,3 1,2 1,3 1,2	② 1,2 1,3	② 1,3 1,2	⑤ 1,3	③ 1,3 1,2	⑩ 1,2 1,2 1,2 1,2	⑴ –	② 1,2	③ 1,2 1,3 1,2	⑤ 1,3 1,2 1,2	⑨ 1,3 1,2	① 1,2	② –	⑥ –	① 1,2	⑪ 1,2 1,2 1,2 1,2 1,1 1,3 1,3 1,2	① –	① 1,2	② 1,1 1,2	
	2	1	1	0	1	4	0	1	2	2	1	1	0	0	1	8	0	1	4	30

FIG. 5 COMPUTER LOCATED INTERSECTIONS BETWEEN WORDS OF RUNNING TEXT AND THREE INDEPENDENT CONCEPTS OF "WHIP"

5B

	Act			Dog		Political	Group	TOTALS
	effect, affect	play role	statute	the animal	to trail closely			
	1?	2#?	3	1#?	2 ?			
Whip1 to move something suddenly	③ 1, 2 1, 2 2	③ 1, 1 1, 3 3	② – 0	⊖ 0	① 1, 2 1	② 1, 3 1, 2 1	⑤ 1, 2 1, 2 1, 3 1, 2 3	10
Whip2 to lash or a lash	③ 1, 3 1, 2 1	③ 1, 2 1, 3 1	② – 0	⊖ 0	① 1, 3 0	② 1, 2 1, 2 2	⑤ 1, 3 1, 2 1, 2 1, 2 1, 2 4	8
Whip3#* political party whip	④ 1, 2 1, 2 2	② 1, 2 1, 2 2	② – 0	⊖ 0	① – 0	② 1, 2 1,1 1, 1 6	⑧ 1, 2 1, 2 1, 1 1, 2 1, 3 1, 1 9	19

	Person	Responsible	See: to view	See: to comprehend	See: to meet with	See: to insure	Political	Group	Parliament	Present: here, now	Present: to put forward, thing given	Desire	Do	Right: correct	Right: opposite left	Right: conservative	Note: musical	Note: record, to record	Note: paper money	Note: voice quality	Request	TOTALS
			1	2	3*	4#				1#*	2			1#*	2	3	1	2#*	3	4		
Whip1 to move something suddenly	③ 1,2 1,2 1,2	⑰ 1,2 1,2	④ 1,3 1,2 1,2	①	② 1,2 1,2	② 1,3	② 1,3 1,2	⑤ 1,3 1,2	⑥ 1,2 1,2	⑤ 1,3 1,1	④ 1,3 1,2 1,2	⑥ 1,2 1,2 1,3	⑩ 1,3 1,2 1,2 1,2 1,2 1,3 1,2	⑦ 1,2 1,2 1,3 1,2 1,2 1,3	—	—	⑤ 1,3	⑩ 1,2 1,1 1,3	①	④ 1,3 1,2 1,3	⑧ 1,2 1,2 1,2 1,2 1,3 1,2 1,2 1,3	39
	3	2	2	0	2	0	1	1	2	3	2	2	5	4	0	0	0	4	0	1	6	
Whip2 to lash or a lash	③ 1,2 1,2 1,2	⑫ 1,2	④ 1,2 1,2 1,3	—	② 1,2 1,3	① 1,2	② 1,2 1,2	⑤ 1,2 1,2	⑥ 1,2 1,2	⑥ 1,2	④ 1,2 1,2 1,3	⑦ 1,3 1,3 1,2	⑩ 1,2 1,2 1,3 1,2 1,2 1,2	⑨ 1,2 1,2 1,3 1,3 1,3 1,2	—	② —	⑦ 1,2 1,2 1,3	—	① 1,2	⑧ 1,2 1,2 1,2 1,2 1,3 1,2 1,2 1,3	34
	3	1	2	0	1	1	2	2	2	1	2	1	5	3	0	0	0	2	0	1	6	
Whip3#* political party whip	③ 1,1 1,2 1,2	⑨ 1,1 1,1	⑦ 1,3 1,3 1,2	—	③ 1,1 1,2	① 1,3	② 1,1 1,1	⑧ 1,3 1,1	⑦ 1,2 1,2	⑥ 1,1	④ 1,3 1,2 1,3	④ 1,2 1,3	⑩ 1,3 1,2 1,3 1,2 1,1 1,2	⑤ 1,3 1,2 1,2 1,1	—	—	① —	④ 1,2 1,2 1,3	—	① 1,2	⑦ 1,3 1,1 1,2 1,3 1,2 1,1 1,3	54
	5	6	1	0	4	0	6	3	2	3	1	1	6	5	0	0	0	2	0	1	8	

	Give			Egg		Quick	Blow		Fork	Get	Mix	Air		Become		Stiff		TOTALS
	to provide	to carry out	to stretch, elasticity	as from chickens	to taunt		to puff, windstorm	Impact				the gas	an appearance	to get to be	to flatter	unbending	alcoholic	
	1#*	2	3	1#	2*		1*	2#				1*	2	1#*	2	1#*	2	
Whip1#* to move something suddenly	⑩ 1,2 1,2 1,2 1,3 1,3 1,2 1,2 1,3 1,2 1,2 7	① 1,2 1	⑤ 1,3 1,2 1,2 1,2 1,1 6	⊖ – 0	⑤ 1,2 1,2 1,2 3	⑧ 1,2 1	⑩ 1,1 3	⑤ – 0	⑫ 1,1 1,2 1,2 1,2 1,2 7	⑦ 1,2 1,2 1,2 1,3 1,2 1,2 5	⑤ 1,2 1,3 1	③ 1,2 1,2 1,2 3	③ – 0	③ 1,2 1	② 1,3 1,3 0	⑦ 1,2 1,2 1,2 1,2 4	① 1,2 1	43
Whip2 to lash or a lash	⑩ 1,2 1,2 1,2 1,2 1,1 1,3 1,3 1,3 1,2 1,2 9	② 1,2 1,2 2	⑤ 1,2 1,3 1,3 1,3 1,1 4	⊖ 0	⑩ 1,2 1,2 1,2 1,2 1,2 5	⑨ 1,2 1	⑩ 1,3 0	⑤ – 0	⑧ 1,2 1,2 1,2 1,3 3	⑧ 1,2 1,3 1,3 1,3 1,2 1,2 3	⑤ 1,2 1,2 2	③ 1,2 1,3 1,2 2	② – 0	③ 1,2 1	① 1,2 1	⑧ 1,2 1,2 1,3 1,2 3	① 1,2 1	37
Whip3 political party whip	⑨ 1,2 1,2 1,3 1,3 1,2 1,3 1,3 1,1 1,2 7	① 1,2 1	② 1,2 1,3 1	⊖ 0	⑨ 1,2 1,2 1,2 1,2 1,2 1,1 8	② 1,2 1	③ – 0	① – 0	⑦ 1,3 1,2 1,3 1	⑥ 1,2 1,2 1,2 1,3 1,2 1,2 5	④ 1,2 1,3 1	② 1,2 1,3 1	① – 0	③ 1,2 1	① 1,3 0	⑤ 1,2 1,2 1,2 1,2 4	⊖ 0	31

5E

	to move, operate	to extend	to become, develop	to be used up, break	normally kept at	to acquire	to accompany in order to move	have desired effect	to react to	to photograph	to require	Quick	TOTALS
	Go					Take							
	1#*	2	3	4	5	1?	2#?	3	4	5	6		
Whip1#* to move something suddenly	⑤ 1, 2 1, 2 1, 2 1, 1 1, 2 7	① 1, 2 1	② 1, 2 1, 3 1	④ 1, 2 1, 2 1, 2 1, 2 4	③ 1, 2 1, 2 1, 2 3	⑨ 1, 2 1, 2 2	③ 1, 2 1, 2 2	③ 1, 2 1	① 1, 2 1	① 1, 2 1	② 1, 2 1	⑧ 1, 1 3	27
Whip2 to lash or a lash	⑤ 1, 2 1, 3 1, 3 1, 3 1, 2 2	① 1, 3 0	② 1, 2 1, 3 1	④ 1, 2 1, 2 1, 2 1, 1 6	③ 1, 3 1, 3 1, 3 0	⑨ 1, 2 1, 2 2	③ 1, 3 1, 2 1	③ 1, 3 0	② 1, 3 0	① 1, 2 1	② 1, 3 0	⑨ 1, 3 0	13
Whip3 political party whip	④ 1, 2 1, 2 1, 2 1, 2 4	⊖ 0	① 1, 2 1	① 1, 2 1	① 1, 2 1	⑥ 1, 2 1, 2 2	③ 1, 2 1, 2 2	③ 1, 2 1	① 1, 2 1	① 1, 2 1	① 1, 2 1	② – 0	15

In Figures 5-a through 5-e these three independent concepts of "whip," the encodings of the Webster's definitions, are represented at the far left of each row. The independent meaning of "whip" which best matches each particular sentence of the text was not indicated to the program, but is indicated in Figures 5-a through 5-e by a symbol # attached to that meaning, for example, to "whip2," in the left-most column of Figure 5-a, since Figure 5-a covers the sentence of text which says, "stick with cord or leather fixed to the end of it ..." In Figure 5-b, "whip3" is marked with a #, and so on through 5-e.

In order to "deepen" the model's knowledge of the various concepts "whip" can refer to, the major content words appearing in each of the three Webster's definitions were also looked up (in the Basic dictionary), and their definitions encoded and added into the memory. This provided a total memory model composed of the definitions of 52 words.

The intersection program was then run repeatedly with this memory base, taking different words of the text as patriarchs. Patriarchs for these runs were selected as follows: In each run, the ambiguous node "whip" was taken as one patriarch. The other patriarch was changed for each run, with the words and word senses shown along the top row of Figures 5-a through 5-e successively being taken as the changing patriarch.

For each such run, therefore, the intersections found constituted a set of connection points between one sense of one word of the sentence, and one or more of the three possible senses of "whip" that the memory had knowledge of. The circled number in the top of each cell in Figures 5-a through 5-e indicates the total number of intersections between the word or

word sense at the head of that column, and the sense of "whip" at the left of that row. Thus there is a set of intersections for each cell.

Then the program also ascertained, for each of the intersection nodes in each such set, whether or not it could also be reached within the concept of any word prior to its column patriarch in that particular sentence. In other words, the program was made to find, for each cell of Figures 5-a through 5-e, an initial set of two-way intersections between a word or word sense of the text and one meaning of whip, and then to select from this set of two-way intersections the subset which were also three-way intersections, the third patriarch being any prior word in the sentence. This technique makes use of syntax only to the extent of assuming that word order progressively adds to the total semantic "context" in which each new word of the text is comprehended.

Every three-way intersection node thus found by the program is represented in the appropriate cell of Figure 5 by a pair of digits.

The second of these digits shows the "depth" of the intersection node beneath the row patriarch, whip1, whip2 or whip3. If this second digit is 1 it means the intersection word appeared directly in the dictionary definition of that particular sense of "whip"; if this second digit is a 2 it means that the intersection word appeared not in the definition of that sense of whip, but in the definition of some word which did appear in that definition of whip; and, if this second digit is a 3 it means that the intersection word appeared in the definition of some word appearing in the definition of some word

appearing in the definition of that sense of "whip." Note that although no third level definitions were specifically put into the memory, an activation search could proceed to that level whenever some word in "whip's" second level happened to appear somewhere else in the Basic text being encoded, and hence had a definition within the memory. Also, although only first level definitions of the text words — the words at the tops of columns — were included, the second or lower levels of their concepts could often be reached by a similar happenstance. However, in Figures 5-a through 5-e intersections beneath the first level for each such word are not tabulated, because beneath the first level of a meaning of any text word, the memory is incomplete, only existing in certain places by virtue of a sort of accident. The same is true beneath the second level in a meaning of "whip," so that for our purpose here we shall not count such intersections, even though intersections at the third level of "whip" are shown by digit pairs in Figures 5-a through 5-e.

The first digit of each digit pair is the depth of that intersection within the concept of the word at the top of the column, and, since others have been excluded, this is always a 1.

Perhaps it should be repeated at this point that only a node constituting a three-way intersection is shown in a cell as a pair of digits. Each digit pair represents one three-way intersection, between: the kind of "whip" shown at the left of the row, the word or word sense shown at the top of the column, and some other prior word or word sense of that sentence. This is why there are usually fewer digit pairs in a cell than the circled number, which represents all the two-way intersections. In the case of the first word of a sentence, there are no prior

words, so in this case it was decided to let all the two-way intersections count, i.e., to consider all these intersections in the same way that three-way intersections of later words were considered. Hence, for each first word of a Figure 5-a through 5-e (except 5-b, see below) all the two-way intersections are shown as digit pairs.

The reason for intersecting words in this rather complex manner is mainly because it can be done very efficiently in a single left-to-right scan, without ever any need for erasing any activation tags, or for retracing over a concept once it has been tagged. That is, the three concepts for "whip" were first activated to level 3, and then each word of the sentence was taken in order and activated. By checking each node reached during this second phase, first for a tag from some meaning of "whip," and, if one was found, then for any tag from any prior word of the sentence, all the tags the program needed to check for were certain to be there by the time they were needed.

Now, how can the intersections discovered be used to disambiguate the words of the text? The first thing needed is some way to weight each intersection according to the depth at which it lies in the patriarch concepts. On this question, all that can be said for certain is that shallower intersections should count more than deeper ones. Thus, we arbitrarily use the scoring key shown in the note for Figure 5. Notice that this assigns all intersections with second digit 3 a zero weight, since these come from an incomplete and fortuitous region of the memory. (Although a scoring key which does count 3rd level intersections doesn't seem to change anything appreciably.) Using the key in Figure 5, a total intersection score for each cell of Figures 5-a through 5-e may be calculated.

This score is shown as the figure in the bottom of each cell. Such a score may be thought of as one measure of the semantic similarity between the word sense at the top of that column and the meaning of "whip" at the left of that row, bearing in mind that this measure is restricted by the prior context established by previous words of the sentence. (And remembering also, of course, that the measure is limited by the fact that the overall memory is relatively underdeveloped, in that it does not permit valid comparative information to be extracted from any level of the memory deeper than 2.) It now can be asked, how valid a measure of semantic similarity the intersection score seems to be, and how it might be used to disambiguate the words of running text.

Consider the situation of a mechanism that "knows" three general meanings for "whip," and that reads a sentence which it knows is talking about one of those meanings but not which one. The mechanism's first job, we surmise, should be to decide which of its three independent whip concepts the sentence is indeed referring to. If all the mechanism can generate is relative measures of the semantic similarities between the meanings of the words of the sentence and each of its independent whip concepts, it could calculate the overall sum of connections between each word's meanings and the three independent "whip" concepts: the TOTAL scores shown in the far right of each row in each of Figures 5-a through 5-e. Selecting the row with the largest TOTAL score, the mechanism could take that row's meaning of "whip" as its guess about the concept this particular sentence pertains to. Recalling that in Figures 5-a through 5-e the correct sense of "whip" for each sentence was marked with a # symbol, the sense with the largest semantic intersection TOTAL within the sentence is marked with a * symbol. We see that for every case, 5-a through 5-e, this judgement is

correct, although for sentence 5-a, it is a very close decision between whip1 and whip2. Note that the program even correctly selects the sense of "whip" that is being discussed in sentence 5-b, even though, as for our coder-subjects, this sentence was misleadingly presented to the program in the context established by sentence 5-a, rather than in that established in 5-c. (After sentences 5-b, 5-c, and 5-d, but not 5-a, the memory was completely "flushed clean" of all activation tags before it went on to process the next sentence.) Hence in the case of sentence 5-b, the program seems to be performing like our subject GB, who noticed the error of putting sentence 5-b in the same context of 5-a, rather than like this author, who made the error, or like subject AX, who failed to catch it.

Once the sense of "whip" being discussed in a sentence has been decided upon, our hypothetical mechanism might next proceed to disambiguate the words of the text. To select one sense of "whip" is to restrict further consideration of intersections to its row in the Figure 5-a, 5-b, etc. Within this row, the mechanism could simply select the largest score out of the cells for each word with multiple meanings, in order to disambiguate it. Along the top of Figures 5-a through 5-e the correct sense for each polysemantic word has again been marked with the symbol #, while the sense indicated by the program's data is marked with the symbol *.

These disambiguations are summarized in Table V. In total, the procedure above gets 12 disambiguations completely correct, three incorrect, and fails to completely come to a decision in four other cases. Figured another way, the program succeeded in eliminating 27 word senses correctly, while only eliminating 3 incorrectly (although this latter measure is positively biased in the case of words with more than two meanings.)

TABLE V

SUMMARY OF COMPUTER DISAMBIGUATIONS

	Sentences of Definitional Text					
	5a	5b	5c	5d	5e	Totals
1. Number of words correctly disambiguated after choosing the correct sense for whip.	4	0	3	4	1	12
2. Number of words left ambiguous after choosing the correct sense for whip.	1	2	0	0	1	4
3. Number of words incorrectly disambiguated after choosing the correct sense for whip.	0	0	1	2	0	3
4. Number of incorrect word senses eliminated.	7	1	8	5	8	27
5. Number of correct word senses eliminated.	0	0	1	2	0	3

At first glance this appears to be an extremely good performance, and was indeed better than we had expected.

However, on closer inspection, the program's output is somewhat less impressive than it initially appears. One notes, first, that the correct sense of each word could have been selected pretty well, merely by always choosing the first sense of each polysemantic word. In that case, one misses in only 6 cases and gets 13 correct, which is not too much worse than the program's 3 misses, 4 left partly ambiguous, and 12 correct.

Also, judging the correct meanings of a word by considering only the correct meaning of "whip," i.e., by considering only scores within that particular row, only constitutes an improvement over selecting the meaning within any other row, in 5 instances, and actually makes things worse in three other instances (either by going wrong, or by introducing ambiguity that is not present in some other row.) In some cases, all this would appear to indicate is that the inappropriate senses of "whip" nevertheless have a good deal of semantic content in common with the appropriate sense. However, in other cases, it would appear that the program's success may result merely from a propensity for picking the most common meanings of words, as for instance, in the case of "stiff" or "right." As a full fledged disambiguator of words in text, then, the program's performance, as shown in Figures 5-a through 5-e, is perhaps not very much above what could be achieved in some much simpler way, although just what this way would be is not clear.

However, while the program seems less than perfect at disambiguating words of sentences on this completely "no-syntax" basis, the Figure 5 data does seem to indicate that the program provides a reasonable, though not very sensitive,

simulation of how one would intuitively feel about semantic similarity. In the first place, the program did indeed get all the meanings of "whip" correct, and, more importantly, the program's scores do seem in general to mirror what one's intuitive feeling would be about the relative similarities of word concepts. For example, in sentence 5-a the program must select whipping something as done with a lash, over whipping something as done with a whisk, and over the political party sort of whip. Between the first two alternatives it gets very close scores, 37 versus 36, but scores whip3, the political party type of whip, only 30. In sentence 5-c, on the other hand, where it must pick out the political party kind of whip from the other two kinds, it gets a strong difference, 54 versus 34 and 39. In general, it would appear that whenever there is a really unequivical distinctness between the meanings it is to discriminate, the program tends to come up with its largest score differences. As another example, consider the three meanings of "stick." "Stick1," which means "some rod-like object," would intuitively seem very closely connected to a whip in the sense of whip2, but not so closely connected to "whip's" other two meanings. The scores in Figure 5-a bear this out, providing 7, 1, 3 in row 2, but 3, 3, 3 and 2, 1, 1 in the other rows.

Even for some of the program's "errors," one would not always want to call the erroneous judgements wrong on purely semantic grounds. For instance, in sentence 5-c, for the word "see" the program judges that a political party-type "whip" has a close connection to "meeting people," but fails to recognize that what is wanted here is the meaning of "see" which means "to insure." Another interesting error occurs in sentence 5-d. Here the program, "knowing" that it is talking about some kind of "whip" or "whipping," decides that "egg" must mean "to egg on," rather than the kind of thing that a

chicken lays. The error is interesting because it would appear to have also been made by subject AX. In paragraph #139 of her protocol, working with "whip," "blows," and "egg," AX temporarily assumes that "egg" is to be a verb.

In most cases where the program's scores appear not to reflect obvious similarities or distinctions between concepts, the definitions provided by the Basic English Dictionary may be the reason for its insensitivity or error, rather than the logic of the memory structure or of the intersection program as such. (For example, the Basic Dictionary defines the meaning of "blow" that has to do with air puffs in relatively complete detail, but dismisses the meaning that has to do with the impact type of blow, "blow2," with only: "Sudden coming against with hand or instrument.") But, a memory model with improved definitions must be left for another project, as of now the model's performance can only be judged by what it can produce, as shown in Figure 5.

In conclusion, it would seem that the attempt made in this section to explain, with the memory model alone and almost no use at all of syntax, how words of text can be disambiguated, has been, as one would expect, less than completely successful. As a measure of intuitive semantic similarity, however, the model would seem to provide a somewhat more reasonable simulation, even though a fairly insensitive one. It does not seem to me that the most promising use of such a memory lies in the sort of blunderbuss approach employed when one simply tallies total numbers of intersections, weighted somehow to reflect their depth within patriarchs. What is needed to approach human performance with such a memory is not gross counts of numbers of intersections, but rather more precise use of the exact nature of the paths which do connect two patriarchs in the

memory. The next chapter, therefore, will consider first how syntactic facets of a sentence seem to have been used by subject AX to provide, to some semantic relating process corresponding to our intersection program, more carefully chosen pairs and trios of "patriarchs." Secondly, Chapter VII will illustrate how paths between patriarchs seem to require judging of a more sophisticated sort than by their "depth" alone, and, following from this, how an activation process may be directed to search selectively through the memory.

CHAPTER VII

DEEPER ANALYSIS: TOWARD A PROGRAM TO UNDERSTAND TEXT

In this chapter an attempt is made to find in the protocol some answer to the main question posed at the end of Chapter V: how does the syntax of a sentence indicate the c[illegible]ations of word concepts to be taken together as patri[illegible]s for intersecting? Although the heart of our proposals for anyone who seriously wants to build a rigorous model of the text understanding process are contained in this chapter, following it will require a considerably deeper immersion in details and in the general theory of Chapter V, than do other chapters, so that the reader whose interest in this particular theoretical issue is more casual may wish to skip to its final three paragraphs.

A. SEGMENTING TEXT BY RULES

In Chapter VI it was noted that AX's first move in encoding is to "bite off" a segment of text for intensive processing. Her segmenting, therefore, constitutes data about the order in which she groups words for intersecting. If formal rules could be stated which would segment the text in the same manner as AX did, these rules would provide some basis for similar sequencing of the intersection process by a computer, and hence would be the beginning of an explicit theory of how text may be understood. Consider the following two rules, which are designed to achieve such segmentation:

The UNIT (U) Forming Rule:

> Proceeding from left to right in the sentence, start the current UNIT after the end of the last UNIT, and extend it until at least one word having at least one meaning that contains no S or D parameter is encountered. Then continue the UNIT (unless a pronoun is a part of it) until a punctuation mark, or any word which has only meanings which contain an S or D parameter is encountered. End the UNIT just before such a word. If a left parenthesis, or the word "that" is ever encountered, recurse, i.e., interrupt the current UNIT formation and process the embedded material completely before completing formation of the current UNIT.[1] (Very roughly speaking, this rule amounts to proceeding on in the sentence until one potential nominal is encountered, and then ending the UNIT in front of the next preposition, certain verb, or punctuation mark.)

Repeated applications of this rule to the text which we have been dealing with produces the "UNITS" labelled U1, U2, etc., beneath row 1 of Figure 4. Now suppose that AX's language processor, after every application of the above rule, and hence after each new UNIT is formed, applies the following rule, which specifies how to form a Verbal Segment:

Verbal Segment (V) Forming Rule

> If the just formed UNIT is to be linked to any node (in the plane as built up until now) that represents a word of the UNIT that immediately preceded the current one (on its same level), then form a Verbal Segment beginning with that word in the proceeding text and extending through the end of the current UNIT. If the current UNIT is to be linked directly to any node not coming from the immediately preceding UNIT, then form a Verbal Segment consisting of the current UNIT alone.

---

[1] AX, puzzlingly, resisted our repeated efforts during training to get her to think of an embedded clause starting with "that" as headed by the main verb of the embedded sentence. Thus, we shall here do it her way, and end such a recursion after the noun and one other UNIT are identified.

This Verbal Segment forming rule cannot be used until the underlined phrase "is to be linked to" is defined. However, the rule can be tested if we define "is to be linked to" post hoc, that is, by whether or not, in the planes that AX in fact created, any direct line connects words whose linking together is in question. Thus, "is to be linked to," will show up in Figure 4 as some line connecting part of the new UNIT to some word in the preceding UNIT, or to some parameter symbol within some such word's concept.

Using this post hoc definition of linkability, and applying the V forming rule immediately after each UNIT is delineated, produces the segments labelled V1, V2, etc., in row 1 of Figure 4.

These V segments form predictions that now may be compared to the actual segments (segs) AX reports aloud in the protocol. This comparison is carried out in Table VI, Parts A through G. In Table VI the columns represent, respectively: (1) the UNITS formed by each application of the U rule, (2) the segments formed by each following application of the V rule, and (3) AX's actual reported segments. A dotted line has been drawn from each predicted segment in column 2 which, in fact, occurs in column 3, that is, for each verified prediction. Since single words read aloud by AX were not eligible to be segs, all single word V predictions are untestable. Thus, in the first sentence, we see that the first prediction, V1, is untestable, the second, V2, is not in fact stated by AX, but that all the remaining seven predictions are matched by segs stated aloud by AX.

For V2, the unconfirmed prediction, AX utters a segment which corresponds to U2, the unit formed by that application

TABLE VI

Comparison of Predicted Segments to Those Produced by AX

Key

Predictions (V's) are marked as:

+ meaning successfully matched by an AX seg.

* meaning unsuccessful but having its UNIT matched.

- meaning neither matched.

meaning an untestable prediction.

6a

n. Stick with cord or leather fixed to the end of it, used to give blows in driving animals, etc., or as punishment.

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U1. stick	#V1. stick	
U2. with cord		S1. "with a cord"
	*V2. Stick with cord	
U3. or leather		
	+V3. cord or leather----	S2. "cord or leather"
U4. fixed to end		S3. "fixed to the end of it"
	+V4. fixed to end-------	S4. "fixed to the end"
U5. of it		
	+V5. end of it----------	S5. "end of stick"
U6. used to give blows		
	+V6. used to give blows----------	S6. "used to give blows
U7. in driving animals		
	+V7. give blows in driving animals----	S7. "give blows in driving animals"
		S8. "give blows"
		S9. "in driving animals"
		S10. "driving animals"
U8. (or) etc.		
	+V8. animals etc.-------	S11. "animals etc."
U9. or as punishment		
	+V9. or as punishment----------	S12. "as punishment"

TABLE VI (Continued)

6b

vt and i. Give blows to with whip
(but AX reads it: "Give blows to something with the whip," see paragraph 45 of the protocol.)

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U10. give blows	-V10. give blows	
U11. to something	-V11. give blows to something	
U12. with whip	+V12. with whip	S13. "with a whip"

Comment: It may be that AX fails to read a segment corresponding to V10 or V11 because she has provided tne word "something."

6c

vt and i. Acting as a whip to dogs or political group.

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U13. acting	#V13. acting	
U14. as whip	+V14. acting as whip	S14. "to act as a whip" S15. "as a whip"
U--- to D(=parenthesized unit)		S16. "to D"
U15. dogs	#V15. dogs	
U16. (or) political group	#*V16. dogs or political group	S17. "dogs or political group"
U17. to(dogs or political group)	-V17. acting as a whip to dogs or political group	

Comment: Here the parentheses in the text function as a signal to recurse within the formation of U14. All such recursions will be indicated by being enclosed in a box.

TABLE VI (Continued)

6d

n. Person responsible for seeing that others of his political group in Parliament are present when desired, do the right thing.

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U18. person responsible		
	+V18. person responsible	S18. "responsible person"
U--- for seeing that		
U19. others		
	#V19. others	
U20. of his political		
	#V20. others of his political group.	
U21. for seeing that others of his political group		
	#V21. responsible for seeing that others of his political group	S19. "responsible for seeing that others of his political group"
		S20. "others of his political group"
		S21. "of his political group"
		S22. "political group"
U22. in Parliament		S23. "in parliament"
	*V22. group in Parliament	
U23. are present		
	+V23. are present	S24. "are present"
U24. when desired		
	+V24. are present when desired	S25. "present when desired"

TABLE VI (Continued)

6e

n. Note from whip requesting person to be present in Parliament, etc.

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U26. note		
	#V26. note	S28. "from a whip"
U27. from whip requesting person		
	-V27. note from whip requesting person	
U28. to be present		S29. "Requests a person to be present"
	+V28. person to be present----------	S30. "For a person to be present"
U29. in Parliament		S31. "is present in Parliament"
	+V29. present in Parliament---------	S32. "present in Parliament"
		S33. "in Parliament"
U30. (or)etc.----------		S34. "or meeting"
	*V30. Parliament or etc.	

6f

vt and i. Give (eggs etc) quick blows with fork etc., to get them mixed with air so that they become stiff.

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U31. Give		
	*V31. Give	
U--- (D = parenthesized unit)		
U32. eggs		
	#V32. eggs	
U33. (or)etc.		
	#V33. eggs etc.	
U34. eggs etc		
	-?V34. give eggs etc.	
U35. quick blows		
	+V35. give (eggs etc) quick blows---------	S35. "giving to the eggs"...quick blows
		S36. "or food"

TABLE VI (Continued)

6f (continued)

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U36. with fork		
	-?V36. with fork	
		S37. "with a fork etc"
U37. (or)etc.		
	+V37. fork (or) etc.-----	S38. "fork etc."
		S39. "or spoon"
U38. to get them		
	-V38. to get them	
U39. mixed		
		S40. "get the eggs mixed with air"
	+V39. to get them mixed----------	S41. "to get them mixed"
U40. with air		
	+V40. mixed with air-----	S42. "mixed with air"
		S43. "with air"
U--- so (following relationship)		
U41. they		
	#V41. they	
U42. become stiff		
	#V42. they become stiff	
U43. so that they become stiff		
	+V43. so that they become stiff---------	S44. "so that they become stiff"
		S45. "so that"
		S46. "the eggs become stiff"

TABLE VI (Continued)

6g

vt and i. Go or take quickly (out, through etc.,)

U-Forming Rule	V-Forming Rule	AX's Reported Segments
U--- go or take quickly (parenthesized unit)		
U44. out	#V44. out	
U45. through (or) etc. (something)	#V45. Out (or) through	
U46. etc.	#V46. out or through or up	
U47. go or take quickly out or through or etc (something)	+V47. go or take quickly out (or) through (or) etc. (something)	S47. "goes quickly out or through something" S48. "out or through" S49. "out or through or up"

TABLE VI (Continued)

6h

Summary of Comparisons of Predictions to Actual Segments:

(1) Sentence	(2) Number of testable predictions (V's inside recursions are not counted)	(3) (hits) Number matched by an AX segment	(4) (near misses) number of remaining predictions for which the U-rule output was matched	(5) (misses) number of predictions for which neither the U or V was matched by AX
a	8	7	1	
b	3	1		2(V10,V11)
c	2	1		1(V17)
d	6	5	1	
e	4	2	1	1(V27)
f	8	5		3(V34,V36,V38)
g	1	1		
Total	32	22	3	7

In addition to the 26 of AX's segments which constitute hits or near misses (Columns 3 and 4, above), she mentioned 22 additional segments. Of these 18 are further breakdowns of longer segments or else matches to something within a recursive unit. Her S3, S29, S37, and S40 are the stretches longer than anything predicted.

of the U rule. She also expresses one long segment, S3, which does not correspond to any prediction of our rules, and three segments, S8 through S10, which appear to be further subdivisions of the long segment V7. One of these, S9, also corresponds to U7.

In later sentences the U forming rule calls for a recursion whenever this is indicated in the text by some indicator such as the word "that" or by a left parenthesis. If this is done the rest of the U's and V's shown in Figure 4 result. Using these to complete Table VI yields the result summarized in Part H of that table. Of 32 testable V segment predictions, 22 are matched directly by AX's segments, and three of the ten remaining V's have their corresponding UNIT matched by one of AX's segments. Two of the remaining seven, V34 and V36, differ from parts of what AX produces only by an "etc." Of the 21 extra segments stated by AX, 16 are further breakdowns of Verbal Segments, or else part of some UNIT embedded in a recursion.

Overall, it appears that the V predictions provide a very good match to the data. This, of course, should not be overly surprising, since the predictions are fairly closely fitted to AX's data by the fact that the actual plane she produced is used in obtaining them. Therefore, showing that these rules predict adequately is only to take a very small step toward a computer program which can simulate AX's coding behavior. However, this step would be much increased if the meaning of "is to be linked to" could be defined independently of any output of AX. Doing so will be considered in Section B, but first, let us consider what the U and V rules imply about the choice and sequencing of patriarchs for an intersection process.

By providing a UNIT, the U rule provides a very small set of words assumed to be closely related. It appears only reasonable that such words should be processed together semantically. In terms of the general theoretical view of Chapter V, this means that intersections between the words of a UNIT are sought, and then that tokens for these words are set up cognitively as a single unified segment of a plane. In doing this, it appears that generally the rightmost word of the UNIT is taken as one patriarch, while the D clues of the word to its immediate left are taken as its opposing set of patriarchs. Assuming that a path is thus found, and that doing so tentatively disambiguates the meaning of the two words, (see Chapter V), token nodes for those words' correct meanings can then be created and linked by an object link ($\searrow$). Exceptions to this procedure have to be made when the word concept on the left has no D parameter (and perhaps in other cases), and it would appear that in these cases the two rightmost words themselves are taken as patriarchs. For example, notice that when AX semantically processes the words of U18, "person responsible," she turns them around to "responsible person," indicating how the link between them is to run. In this case, apparently, the intersection path found between the two words taken as patricarchs has determined what the link between their tokens is to be. In the case where there are more than two words in a UNIT, we assume intersection is continued and tokens further formed and interlinked, until the complete UNIT has been represented by a section of a plane.

It should be noted that application of the U forming rule stated above sometimes produces a large UNIT like U27: "from whip requesting person." The fit of the predictions to AX's segments could be improved at certain points if UNITS such as U27 were broken up. It isn't clear whether this indicates that the U rule should have a more lenient terminating condition, viz.,

whenever a possible (rather than certain) verb is encountered, or that it indicates that sub-UNITS should be formable within a UNIT after it has been delineated by the U rule. In AX's data, terminating UNITS in this more lenient way makes the predicted V segments deviate further from her reported segments, since it splits UNITS such as U6, "used to give blows" into two UNITS. (Since "give" can be a noun and "blows" can be a verb. It might be possible to keep U6 a single UNIT by including a rule stating that the word "to" makes "give" necessarily a verb, but consider: "the tightrope artist had trouble walking across the rigid bar, because he was used to give in a high wire.")

Another way to improve the fit of V predictions to AX's segments is to interpret more of the text as falling into embedded units, e.g., the remainder of the sentence following "requesting" in U27. However, we see no way to do this by a straightforward rule, and, more importantly, it just doesn't appear to be the way in which AX understands such constructions.

Whatever may be the case for such details, let us assume that some subplane of tokens somehow does get created to represent each delineated UNIT, and move on to the next question facing the reader: how that subplane is then to link to the prior structure of tokens already formed to represent text prior to the current UNIT.

B. LINKING A CURRENT UNIT'S TOKEN REPRESENTATION TO THE PLANE OF TOKENS REPRESENTING TEXT PRIOR TO IT

The processing involved in deciding how to link a UNIT'S subplane to the overall plane of its text involves many factors, and we shall have to confine our discussion here to the one most dependent upon semantic memory. It is at this step of processing

that searches for intersections within the semantic memory can clearly be seen to be restricted, such that only certain paths between concepts are acceptable. The way in which this occurs will be seen most easily with an example.

Consider the sentence,

"I saw the dirt from his farm on his sleeve."

Suppose a person reading this sentence has already read everything up through the word "farm," and has cognitively established a plane of tokens to represent that much of the sentence. Then, this reader's equivalent of our U rule operates, providing him with the UNIT "on his sleeve." Suppose he then succeeds in intersecting its words to each other, and after that creates a subplane of tokens to represent the phrase. Now, his problem is how that subplane should be linked to the rest of the plane for the preceding part of the sentence. The "correct" way for the reader to establish this link, of course, is to make the phrase a modifier of "dirt," since this "dirt," but not "farm," is likely to be "on his sleeve." Notice first that this decision cannot be decided by any general syntactic rule, since then a sentence like "I saw the dirt from his farm on the Hudson" would be parsed and understood incorrectly. In this case it is the farm that is on the Hudson, not the dirt. Similarly, we see no way to formulate general parameter clues or "selectional restrictions" that would resolve this problem. However, a very general way of making such decisions correctly can be specified, if a semantic memory is assumed.

Suppose that the reader's text processor fires an expanding sphere of activation from the noun in the new UNIT, "sleeve" and also fires a sphere from the immediately preceding noun,

"farm." The full concepts of "farm" and "sleeve" are found to intersect, let us say, at the word "cotton." The path from "farm" through "cotton" to "sleeve" amounts to the information that: "a farm may produce cotton, and a sleeve can be made from cotton."

Now, a subroutine checks to see if the relationship between "farm" and "sleeve" that is expressed by this complex path is compatible with any meaning of "on." In other words, this subroutine is a sort of path-mapper, which will, in general, tell us whether one given path (complex relationship) can without distortion be subsumed by a given single word relationship like "on."

Its answer in this case must be "no," telling the executive language processing routine, in effect, that there is no positive reason to believe that a farm can be on a sleeve. So the hypothesis that "on a sleeve" modifies "farm" is temporarily rejected. Perhaps another intersection between these two words would be found in the semantic memory, and similarly rejected because it was not describable by the word "on."

After these possibilities are rejected, an intersection between "sleeve" and the noun preceding "farm" in the sentence, viz., "dirt," is sought. This intersection occurs, let us say, at the word "stain," and the path through it reads something like, "dirt can be a stain and a stain is something in cloth and a sleeve can also be cloth." Again this path is tested for its compatibility with the relation "on." By rules not very much more complex than those now in our sentence generating program, this path could be cancelled to "dirt can be in a sleeve." This path is logically compatible with one sense of the word "on." Therefore the path-mapper routine should report,

in effect, "yes, dirt can be on a sleeve." Hence, "on the sleeve" would tentatively be judged to modify "dirt."

If, as specified, the full sentence was, "I saw the dirt from his farm on his sleeve," this judgement would be correct and the sentence would be parsed correctly. However, the possibility that the sentence also might have been, say, "I shoveled the dirt from the farm on his sleeve," indicates that the method we have outlined for solving this problem illustrates, at best, only one routine that must be embedded within a larger program which takes other factors into account.

Had the sentence been, "I saw the dirt from his farm on the Hudson," the same semantic match procedure, assuming it were to work correctly, would decide that "on the Hudson" modifies "farm" not "dirt."

The comparison of these two cases also illustrates the futility of attempting to perform such differential parsings on the basis of pseudo-syntactic classifications (what Katz and Fodor call "semantic markers"), since these are not nearly as rich as the clue words of this model, and clue words will not work in the above case. The crucial factor in the decision as to what the final prepositional phrase is to modify is how its full semantic concept interacts with those of preceding nouns or verbs. We do not see how any rule independent of a large semantic memory is going to be able to decide such a question. In AX's protocol, the decision procedure outlined above should come into play every time AX decides how a prepositional phrase is to be attached to the structure she has constructed up to that point. In other words, whenever an MB(n) occurs in the protocol, AX has had to decide what some prepositional phrase modifies.

Perhaps a better way to model the processing involved above would be to somehow use the preposition, in this case "on," as a restriction on the activation or search process itself. Then only acceptable intersections would ever be found, whereas, in the process as outlined above, intersections are simply located without restriction, and then each one is tested for its compatibility with the stated relationship, "on." In the next chapter a change in the memory structure will be discussed which would make such "guided" activation of a full concept easier.

In any case, it is because we see prepositions functioning in the above manner, as restrictions on the type of relationship that can exist between other concepts, that we did not, in Table III and in Figure 4, set up as a choice confronting the subject the various meanings which dictionaries list for prepositions. After reading a sentence the understander can often state what its prepositions mean, but his knowledge then is likely to either be very much more or much less precise than that given in dictionaries. According to the account developed above, this is because his understanding of the meaning of prepositions comes essentially out of inferences from his prior knowledge, rather than from information stored with the preposition itself. This phenomenon is also illustrated in the case mentioned in the first chapter, in which a dictionary says the word "the" can mean "her." People reading this definition may be puzzled until an example is given, such as "I took my wife by the hand." What is here being proposed is that the concept "her" is not so much introduced by "the," but rather is allowed by it. Given the example sentence a reader intersects the concepts "hand" and "wife" in his semantic memory, gets "her" as their discovered relationship, and concludes that "the," in this case, means "her." This view of the role of prepositions

and articles (and perhaps other words), was forced to our attention in trying to suffix a number to the words we coded to show their appropriate meanings in definitional text. The particular dictionary definition that is most appropriate for a noun or verb in some given sentence is usually quite easy to select, while that of a preposition is often very difficult, and unsatisfying after it is chosen.

In summary, while this analysis of AX's protocol still leaves many unanswered questions about her processing of text, enough does seem to be observable in the protocol to lay some groundwork for a program to simulate such behavior. The most obvious questions that have been skipped over perhaps are how AX identifies the referents of anaphoric words (see Olney, 1965) and how she performs whatever transformations are necessary to understand text (see, e.g., Petrick, 1965). Despite these omissions, however, an overall picture of the process of understanding emerges from the analysis above, and this picture is sufficient to provide some answer to the question posed initially: just when and how during the understanding of text does the understander rely on semantic facts, and when and how does he rely on syntactic ones?

The process of understanding implied by the above analysis, and based, we emphasize, on the assumption that to understand text is to build up mental configurations of token nodes which represent the text's meaning, may be summarized as follows: As he first identifies the words of a sentence, the understander carries out a step corresponding to what we have called the UNIT forming rule. This rule "bites off" a segment of text for intensive processing. As formulated above, this rule is purely syntactic; only the parameter symbols available in words are needed to apply it.

As each unit is thus delimited, its words or clues are intersected with each other in memory, and, assuming appropriate paths can be found, a conceptual configuration of tokens is formed to record how that UNIT's words interrelate. Then the way that this sub-configuration is to be linked to the previous token structure representing the text must be decided. How this decision is made is unclear, but appears often to depend on finding in memory semantic paths which are compatible with relationships stated by a preposition, article, or other word.

The decision as to how the subplane of tokens representing a just-formed UNIT is to be linked to the prior structure of tokens also seems to get reflected aloud in a verbal "segment" if the understander happens also to be generating a verbal protocol of his processing. It is by looking at these segments for AX, that we have attempted to get some hint about how her sequencing of semantic intersecting may proceed. After the new UNIT's representation is linked to the prior structure, the encoding of one UNIT is thus completed, and another can be "bitten off" for similar semantic processing and representation.

CHAPTER VIII

SOME FINAL IMPLICATIONS AND RELATIONS TO LINGUISTIC THEORY

In this work a tentative theory of the general structure of long-term memory has been developed, and explored in three ways: First, a model of such memory has been utilized in a computer program to simulate human performance on a semantic task. Second, the memory model has been combined with the techniques of protocol collection to produce a methodology for gathering data relevant to how a subject understands text. Third, this subject's internal semantic memory has been assumed to be structured and used as is the memory model, and the implications of this assumption have been utilized to explain the subject's performance, and to develop a tentative theory of how text is understood.

In this chapter some findings and implications of the research will be drawn together. In Section A findings by hindsight about how the memory should have been built in the first place, and how it will be built in future research, will be considered. In Section B implications for the relationship of transformational grammar to psychological performance models will be discussed. Section A is concerned with rather technical details of the model, Section B with very global considerations.

A. IMPROVEMENTS OF THE MODEL[1]

In the model as presented so far, modification of a concept has been encoded by attaching a link labeled "modifier" to a token for the concept to be modified. This modifier link leads to some other node which forms (the head of) the modifying structure. (In IPL terms, the token node is given an attribute whose name is "modifier," and whose value is the top node of the modifying structure.) In a case where the modifying structure is a prepositional phrase modifying a noun or verb, it now appears that it would be much better to simply label the modifying link with the preposition itself, and run this link from the token node to be modified to the object of the preposition. This would eliminate the link labeled "modifier," and hence would reduce the size of the over-all model. More importantly, it would permit a search (activation of a full concept in the memory) to be directed more readily. One need for such directed searches was described in regard to the parsing of prepositional phrases in the last chapter.

In general, it appears that labeling links with words themselves, and especially with prepositions, instead of with predefined linkage names such as "modifier," is a development worth further exploration. The use of prepositions to label links between tokens in the model would seem to go along with another change, the need for which is pointed up by a recent paper by Fillmore (1966). Fillmore's work, although motivated by purely grammatical considerations, indicates convincingly that the parameter symbols S, D, and M, as conceived in the model so far,

---

[1] Alterations discussed in this section are being incorporated into a new version of the memory model and program now being developed in conjunction with Dr. D. G. Bobrow.

are inadequate to achieve the kind of "carrying" of information into alternate forms that is discussed in Appendix III. Briefly, Fillmore's examples raise issues of the following sort: Suppose the verb "to swarm" were given the definition: "for (bees, ants, etc.) to cluster in some area." As this definition has been encoded, the parenthesized phrase becomes a parameter symbol, S, representing whatever the subject of "swarm" is in some sentence or plane where it occurs. Suppose, however, that we now encounter the sentence: "the gardens swarm with bees." If the subject of "swarm" in this sentence is treated as a value for the parameter symbol S, it will mean that, "the gardens cluster in some area." This misinterpretation is due to the fact that S is too gross and undifferentiated a notion, and Fillmore's examples indicate that our S's, D's, and perhaps M's must be subdivided into more precise categories. Fillmore proposes syntactic terms such as "ergative," "agentive," and "locative," as more precise categories which should replace the grosser notions of subject (S) and object (D).

We know (see Appendix III) that the kinds of parameter symbols utilized in the memory must have a clearcut correspondence to the kinds of intertoken links used in it. It would appear that using prepositions to label intertoken links refines the memory's ability to differentiate these relationships in a way that matches the more differentiated parameter symbols that Fillmore shows a need for. In this regard, it appears that, whenever possible, prepositions should be used as links between verbs and their subjects and objects, thereby replacing (ν_1) and (ν_2). For example, the definition of "swarm" given above might be rewritten before encoding to say: "'swarming' is clustering in (some area) by (bees, ants)." During encoding, both parenthesized phrases in this definition would then become parameter symbols, perhaps called, respectively, the "ergative" (E)

and the "locative" (L). Then the sentence, "the gardens swarm with bees," would be interpreted by taking "gardens" as the value of L, rather than of the grosser S. Once this was done, the kind of misinterpretation described above would be avoided.

To refine the memory by using more differentiated inter-token links and parameter symbols is to move it to a level of specificity one step further from natural language than it has been. (And from the usual specificity of transformational structures; as one questioner at Fillmore's presentation put it, his examples concern the "deep, deep structure" of language.) To move the code further from natural language would appear to put more burden on whatever processes translate back and forth between the model's representation and natural language, e.g., the sentence producing routines of our Chapter III program, and the coder (or eventual program) that encodes textual material into the model's representation. However, the opposite may in fact be the case; Fillmore suggests what may be more general generation rules based on his "deep, deep" structures, and, on the encoding or "understanding" side, more precisely differentiated parameter symbols may be easier to select values for than are the grosser categories S, D, and M. (For example, for the sentence above, consider intersecting "garden," taken as one patriarch, with the total set of clue words: "Bees," "ants," and "area," taken together as the other patriarch set. It would appear that an intersection program could easily select the matching member of this set, "area," and hence the correct parameter for "garden" to fill. Turning the sentence around ("bees swarm in the garden,") would not effect the "understanding" achieved, and would hardly even change the process by which it was achieved, since the essential part of this process is the semantic intersecting, rather than syntactic analysis.)

Another change in the model relates to its ability to represent ambiguity easily. A coding convention which the reader may have noticed to be a departure from ordinary grammatical procedure was to make some prepositions which modify a verb the object of that verb. This was done to allow indirect objects and other nouns to be made the subject of the preposition, and hence permit certain fine distinctions of meaning to be encoded. For example, consider a sentence like, "I threw the man in the ring." This sentence can mean: (a) "while in the ring I threw the man," (b) "I threw the man who was in the ring," or (c) "I threw the man into the ring." The encodings corresponding to these three meanings are shown in Figure 6, parts A, B, and C, respectively.

Presented in this way the distinctions of meaning between (a), (b), and (c) are clear, and the encodings logical, even though using a prepositional phrase as an object of a verb (in c) is contrary to usual practice. However, when such a sentence is encountered in text it is often impossible to decide which of its meanings is intended. It turns out, moreover, that coders are most unreliable and unhappy about making this distinction, even in cases in which one meaning does seem clearly indicated.

Thus it would appear that our coders, at least, mentally encode most cases of this kind in some form that leaves it ambiguous as to which exact meaning is intended. In order to be ambiguous on this matter in the code as it stands the coder must set up all the alternate forms, A, B, and C, and then group these into a disjunctive set. In the program now under development all meanings strictly like (c) will be encoded in the form of A, while all meanings strictly like (a) will be encoded as modifiers of the subject ("I" in the above example). This

"I THREW THE MAN IN THE RING"

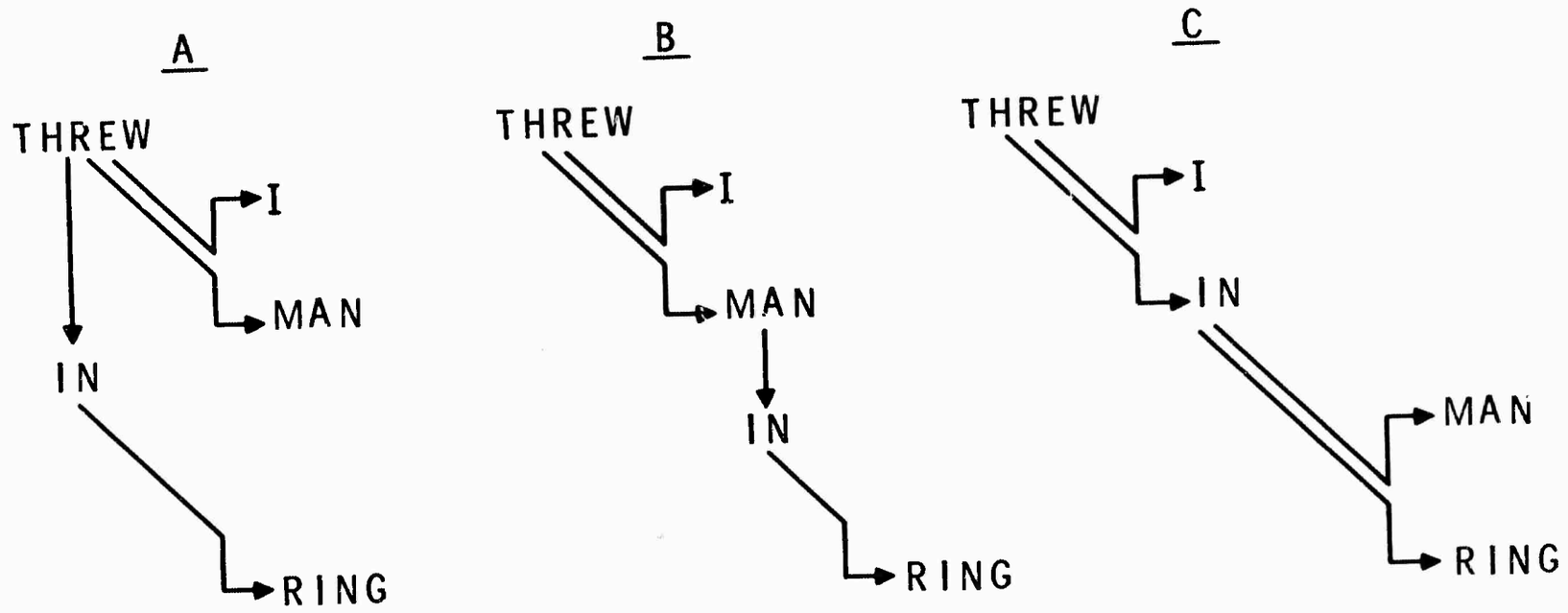


FIG. 6 ENCODINGS OF THREE MEANINGS OF A SENTENCE

eliminates forms of the type C and brings the code more in line with standard terminology. More importantly, however, a new parenthesis notation is being added which will allow a phrase like "in the ring" to simply modify the entire remainder of the sentence, while remaining uncommitted as to just which sub-element it modifies, and hence which precise interpretation is intended. Thus the ambiguous form will be easily represented, while the finer distinctions will require slightly more structure. This clearly will increase the psychological verisimilitude of the model, as well as provide a more useful representation. Considerations of this kind, incidentally, cast doubt on efforts to recode natural language into symbolic logic, and also suggest that programs attempting to process natural language have their greatest hope of success if they are kept as close as possible to human methods and representations.

B. IMPLICATIONS FOR THE RELATIONSHIP OF TRANSFORMATIONAL GRAMMARS TO PSYCHOLOGICAL PERFORMANCE MODELS

The viewpoint that emerges from this research conflicts at several points with the relationship that has been said to exist between current linguistic theories and performance models, especially as these involve semantics. The rest of this chapter will be devoted to clarifying this relationship, as it appears once a semantic memory is assumed to be part of the mechanism or organism that deals with natural language.

In the first place, we simply do not believe that performance theories or computer models can ignore or put off semantics, as most language processing programs so far have done, and yet hope to achieve success. Whether a program is intended to parse sentences, to translate languages, or to answer natural language questions, if it does not take account of semantic facts both

early and often, I do not think it has a chance of approaching the level of human competence. Correspondingly, any theory of language, such as that prescribed by Katz and his co-authors, which asserts that semantic processing is in any sense temporally or logically subordinate to syntactic processing, would seem to be of very dubious value, at least to anyone interested in performance models.

Secondly, it has already been pointed out that the normal process of understanding language, as we see it, need not refuse to process any sentence because the sentence is "semantically anomolous." This is because the memory model provides a natural measure of the relative semantic similarity between one full concept and any others, and hence allows an understanding process to select the best available interpretation for any given word string, instead of first insisting that that string meet previously anticipated conditions in order to be semantically interpretable. Viewing the process of language understanding in this way eliminates the embarassing necessity to say that people must interpret an "anomolous" sentence by some mysterious process based on the sentence's "direct analogy to well-formed sentences" (Chomsky, 1965, p. 149). Positing such a semantic memory thus makes the judgement as to what is and what is not a semantically anamolous sentence arbitrary — as, incidentally, it has always seemed to many people anyhow: consider, for example, Katz and Postal's assertion that the sentence, "the paint is silent," cannot be understood by the normal rules of language interpretation (1964, p. 25).

It has also been noted above that positing a semantic memory seems to abrogate the need in a performance model for the phrase structure component of a transformational grammar, that a set of rules corresponding to the transformational component is

all that our program employs to generate sentences.

Another of the unquestioned tenets of transformational linguistics is that a single grammar should be considered to underlie both the production of sentences and the understanding of sentences. This notion, if feasible, would provide a great simplification of abstract linguistic theory, and seems to be almost universally accepted among transformational linguists (Note, e.g., Chomsky's constant reference to an undifferentiated "speaker-hearer," 1965).

However, it seems clear that while generative grammars are very natural parts of a sentence production mechanism, these same grammars raise immediate problems when one attempts to base a parsing or understanding program on them. To parse according to such a grammar must involve a series of trial matching operations, and the tree of possible matches, even if pruned by heuristics as suggested in Chapters VI and VII, seems inevitably destined to make it more difficult to parse, and hence to understand sentences with any given grammar, than it is to produce them with that same grammar.

This is blatantly opposite to the facts about people: a child can understand more complex sentences than he can generate, a student in a foreign language finds it easier to understand or read correct sentences than to speak them, and a person can read language faster than he can compose it.

In spite of these contradictory facts, the assumption that a single grammar is the best way to explain both the competence of a speaker and of a hearer is widely held. In order to explain a grammar's use in understanding, Miller and Chomsky (1963, p. 465) have adopted a version of the "analysis-by-

synthesis" theory apparently first put forward by Halle and Stevens for phonemics (1959, see also Matthews, 1961).

The key assumption of the analysis-by-synthesis theory is that, in order to understand language, one essentially must re-create the generation process by which that language was created. This re-creation is thought to rely upon the grammar, and to be guided by cues in the given text. The re-creation is continuously checked and corrected by testing tentative steps of generation against further text. Hence, from this viewpoint, to understand text is to locate all the steps that, given the same grammar, might have generated it.[1]

This analysis-by-synthesis model follows naturally from assuming that all language competence is to be explained in terms of a single basic set of rules to be called the grammar. On the other hand, if the understanding of text is viewed as the creation of some mental symbolic representation, such as that comprising the memory model, then there is little reason to suppose that a reader or hearer must retrace the steps by which a sentence might have been generated in order to understand it. The relationship between producing and understanding a given piece of language lies only in the single message content underlying both, not in the processes for moving between English and that message content.

---

[1] From this sort of conception Chomsky has been led to a heavily a priori, anti-environmental theory of language acquisition (1965). His reasoning seems to be: A person can understand a very wide range of sentence structures, yet he can only understand what he also could have generated. Therefore, the person must be born with a very high-powered but latent generation grammar, and must somehow be able to actualize a latent rule of this grammar whenever it is needed in order to understand some sentence containing an unfamiliar syntactic complication.

For instance, suppose a bundle of facts x_1, x_2,...,x_n modify object y. Speaker A knows this, and wishes to tell hearer B. To do so he must utilize some way of turning the conceptual connections which represent this information in his head into natural language. In doing so he has many choices: how many sentences to use; which x's to make into adjectives of y, which ones into predicate nominatives, and which ones into prepositional phrases; whether to use passive sentences, subordinate clauses, rhetorical questions, etc., etc.

However, to obtain an adequate conception of A's meaning, B only needs somehow to arrive at a mental state in which all the x's are represented in his cognitive representation, and are linked as modifiers to a representation of y. The question of how A happened to express each of these facts is of no necessary concern to B; all he must obtain is some mental representation corresponding roughly to that which A is trying to communicate.

Since much of the information that is in A's language therefore can be ignored by B, the tests that he must apply to extract from A's speech what he needs to know clearly can be much simpler than the tests he would have to apply to fully regenerate B's sentence generation process. In other words, it would seem, both from thinking about the problem in terms of a semantic memory model, and from the obvious facts about the relative difficulty, for people, of understanding language versus producing it, that the process by which a person understands language is most likely "primary," in the sense that it does not rely on any generative grammar or sentence production process. Instead of always understanding language by relating it to how a generation grammar would allow the same text to be produced, the understanding process is an autonomous process of its own.

To view a person's understanding of language as a separate problem, independent of any generation grammar, is of course not to say that an understander can ignore the facts of sentence structure — although people can in fact understand text whose sentence structure is incorrect, wildly distorted, etc. — but is only to say that a generative grammar need not be in any sense a "component" of the understander's language processor.

Once the domain which a generative grammar has to account for is thus restricted to sentence production, it becomes unnecessary to think of the grammar as a single set of rules that will generate all constructions of English. Nine years after the publication of Syntactic Structures (Chomsky, 1957) no one has yet succeeded in building a general generative grammar for all of English. This alone is no condemnation of his proposals, but it does seem much more reasonable to hope for a grammar which is capable of accounting for how any given idea may be expressed in some stretch of acceptable English text. Then, to get this grammar to express the same thought in some other style, i.e., in different sentence structure(s), a higher level rule could perhaps be written which would alter the grammar itself. This would provide a mechanism similar to a person, in that it could express itself in text of some style, and perhaps change that style for various occasions, but would not be a mechanism which would simultaneously contain rules capable of producing English of all possible styles.

The situation here would seem to parallel that for speech accents. Many Americans can approximate a Southern accent, or an Irish or German or French one, but no one can simultaneously speak with all these accents. The task of formulating a single phonological grammar adequate to generate speech in all accents at once clearly is unnecessarily difficult, if

even possible. Similarly, it seems unreasonable to seek a grammar that would generate English sentences of all possible styles at once.

Note that the requirement that a generative grammar be universal across all of a natural language's styles of expression cannot be escaped as long as the grammar is considered to underlie sentence understanding as well as sentence production, for an intelligent native speaker has the competence to understand almost any grammatically acceptable sentence style. However, this situation changes once the grammar's job is restricted to language production, and language understanding is attacked as a separate problem.

In summary, therefore, the implications of assuming a semantic memory for what we might call "generative psycholinguistics" are. (1) that dichotomous judgements of semantic well-formedness vs. anomaly are not essential or inherent to language performance; (2) that the transformational component of a grammar is the part most relevant to performance models; (3) that a generative grammar's role should be viewed as restricted to language production, while (4) sentence understanding should be treated as simply a problem of extracting a cognitive representation of a text's message; (5) that until some theoretical notion of cognitive representation is incorporated into linguistic conceptions, they are unlikely to provide either powerful language processing programs or psychologically relevant theories.

Although these implications conflict with the way others have viewed the relationship of transformational grammars to semantics and to human performance, they do not eliminate the importance of such grammars to psychologists, an importance

stressed in, and, indeed, largely created by, the work of Chomsky. It is precisely because of a growing interdependence between such linguistic theory and psychological performance models that their relationship needs to be clarified.

BIBLIOGRAPHY

Banerji, R. B. A language fc the description of concepts. Unpublished dittoed paper, Systems Research Center, Case Institute of Technology, 1964.

Bartlett, F. C. Remembering, a study in experimental and social psychology. Cambridge, England: Cambridge University Press, 1932.

Baylor, G. W., and Simon, H. A. A chess mating combinations program. Proceedings of Spring Joint Computer Conference, 1966, 28, 431-447.

Berkeley, E. C., and Bobrow, D. G. The programming language LISP: its operation and applications. Cambridge, Massachusetts: Information International, Inc., 1964.

Bobrow, D. G. Syntactic analysis of language by computer — a survey. Proceedings of the Fall Joint Computer Conference, 1963, 24, 365-387.

Bobrow, D. G. Natural language input for a computer problem solving system. Unpublished PhD. dissertation, M.I.T. 1964. Also Project MAC, Report #TR-1, 1964, Cambridge, Mass.

Bobrow, D. G. and Teitelman, W. Format-directed List Processing in LISP. Cambridge, Massachusetts: Bolt Beranek and Newman Report #1366, 1966.

Bruner, J. S. On perceptual readiness. Psychological Review, 1957, 64, 123-152.

Bruner, J. S., Goodnow, J. J., and Austin, C. A. A study of thinking. New York: John Wiley and Sons, Inc., 1956.

Bruner, J. S. and Minturn, A. L. Perceptual identification and perceptual organization. Journal of Genetic Psychology, 1955, 53, 18-21.

Chomsky, N. Aspects of the theory of syntax. Cambridge, Massachusetts: The M.I.T. Press, 1965.

Chomsky, N. and Miller, G. A. Introduction to the formal analysis of natural languages. In D. R. Luce, R. R. Bush, and E. Galanter (Eds.), Handbook of Mathematical Psychology, Vol. II. New York: John Wiley and Sons, 1963.

Chomsky, N. Review of Skinner, B. F. Verbal Behavior. Language, 1959, 35, 26-58.

Cliff, N. Adverbs as multipliers. Psychological Review, 1959, 66, 27-44.

Creelman, M. B. The experimental investigation of meaning. New York: Springer Publishing Company, 1966.

Darlington, J. Translating ordinary language into symbolic logic. Memorandum MAC-M-149, Project MAC, M.I.T., Cambridge, Massachusetts, 1964.

Deese, J. On the structure of associative meaning. Psychological Review, 1962, 69 (3), 161-175.

Ervin, S. M. Changes with age in the verbal determinants of word association. American Journal of Psychology, 1961, 74, 361-372.

Feigenbaum, E. A., and Simon, H. A. Performance of a reading task by an elementary perceiving and memorizing program. Behavioral Science, 8 (1), 72-76, 1963

Feigenbaum, E. A. An information processing theory of verbal learning. Report P-1817, the RAND Corporation, Santa Monica, California, 1959.

Feigenbaum, E. A. and Feldman, J. Computers and Thought, New York: McGraw Hill, 1963.

Fifth Annual Report, the Center for Cognitive Studies, 1964-65. The Center for Cognitive Studies, Harvard University, Cambridge, Massachusetts, 1965.

Fillmore, C. J. A proposal concerning English prepositions. Paper presented at M.I.T., Cambridge, Massachusetts, April, 1966.

Funk and Wagnall's new "standard" dictionary of the English language. New York: Funk and Wagnalls Company, 1959.

Gelernter, H. Hansen, J.R. and Loveland, D.W. Empirical explorations of the geometry-theorem proving machine. Proceedings of the Western Joint Computer Conference, 1960, 17, 143-147.

Green, B. F., Wolf, A. K., Chomsky, C., and Langhery, K. Baseball: An automatic question answer. Proceedings of the Western Joint Computer Conference, 1961, 19, 219-224.

Halle, M., and Stevens, K. N. Speech recognition: A model and a program for research. In Fodor, J. A., and Katz, J. J. (Eds), The Structure of Language: Readings in the Philosophy of Language. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1964, 604-612.

Hays, D. G., (Ed). Readings in automatic language processing. New York: American Elsevier Publishing Co., 1966.

Hunt, E. B. Concept learning: an information processing problem. New York: John Wiley and Sons, Inc., 1962.

Katz, J. J. and Foder, J. A. The structure of a semantic theory. Language, 1963, 39, 170-210.

Katz, J. J. and Postal, P. M. An integrated theory of linguistic descriptions. Cambridge, Mass. The M.I.T. Press, 1964.

Kelly, G. The psychology of personal constructs: Volume I. New York: W. W. Norton and Company, Inc., 1955.

Klein, S. Automatic paraphrasing in essay format. SP-1602/001/00, System Development Corporation, Santa Monica, California, 1964.

Klein, S. and Simmons, R. F. Syntactic dependence and the computer generation of coherent discourse. Mechanical Translation, 1963, 7(2), 50-61.

Kuno, S. K. Multiple-Path Syntactic Analyzer. Mathematical linguistics and automatic translation, Report No. NSF-8, 1963, Computation Laboratory of Harvard University, Cambridge, Massachusetts.

Kuno, S. K. The predictive analyzer. Communications of the Association for Computing Machinery, 1965, 8 (7), 453-462. Reprinted in Hays, 1966.

Lakoff, G. On the nature of syntactic irregularity. Mathematical Linguistics and Automatic Translation. Report No. NSF-16, 1965, Computation Laboratory of Harvard University, Cambridge, Massachusetts.

Lamb, S. The sememic approach to structural semantics. In K. A. Romney and R. D'Andrede (Eds), Transcultural studies in cognition. American Anthropologist, 1964, 66 (3), Part 2.

Lane, H., and Schneider, B. Some discriminative properties of syntactic structures. Journal of Verbal Learning and Verbal Behavior, 1963, 2(5-6), 457-461.

Lindsay, R. K. Inferential memory as the basis of machines which understand natural language. In Feigenbaum, E., and Feldman, J. (Eds.) Computers and Thought. New York: McGraw Hill Books Co., 1963, 217-233.

Matthews, G. H. Analysis by synthesis of sentences of natural languages. Proceedings of 1st Int. Cong. on machine translation of languages and applied language analysis, 1961, Teddington, England: National Physical Laboratory.

McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P. and Levin, M. I. LISP 1.5 Programmer's Manual, Cambridge, Massachusetts: the M.I.T. Computation Center, 1962.

Miller, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review, 1956, 63, 81-96.

Miller, G. A., and Chomsky, N. Finite models of language users. In Luce, R. D., Bush, R. L., and Galanter, E., (Eds). Handbook of mathematical psychology, Volume II, New York: John Wiley and Sons, Inc., 1963, 419-488.

Miller, G. A., Galanter, E., and Pribram, K. H. Plans and the structure of behavior. New York: Holt, 1960.

Minsky, M. Steps toward artificial intelligence. Proceedings of the Institute of radio engineers, 1961, 49: 8-30.

Minsky, M. A selected descriptor-indexed bibliography to the literature on artificial intelligence. In Feigenbaum, E. A., and Feldman, J. (Eds.) Computers and Thought. New York: McGraw-Hill Book Co., Inc., 1963, 453-456.

Newell, A. (Ed.) IPL-V programmer's reference manual. Memorandum RM-3739-RC, The RAND Corporation, Santa Monica, California, 1963.

Newell, A., Shaw, J. C. and Simon, H. A. The processes of creative thinking. In H. E. Gruber, G. Terrell, and M. Wertheimer (Eds.) Contemporary approaches to creative thinking. New York: Atherton Press, 1962, 63-119.

Newell, A., Shaw, J. C., and Simon, H. A. Chess playing programs and the problem of complexity. I.B.M. Journal of Research and Development, 1958, 2, 4, 320-335.

Newell, A., and Simon, H. A. Computers in psychology. In R. D. Luce, R. Bush, and E. Galanter (Eds.), Handbook of Mathematical Psychology, Volume 1. New York: John Wiley and Sons, 1963, 361-408.

Newell, A., and Simon, H. A. An example of human chess play in the light of chess playing programs. Pittsburgh, Pennsylvania: Carnegie Institute of Technology, 1964, (dittoed).

Ogden, C. K. The General Basic English Dictionary. New York: W. W. Norton and Company, Inc., 1942.

Olney, J. Building a concept network for retrieving information from large libraries: Part I. Technical Memorandum TM-634/001/11, System Development Corporation, Santa Monica, California, 1962.

Olney, J. C. Some patterns observed in the contextual specialization of word senses. Information Storage and Retrieval, 1964, 2, 79-101.

Osgood, E. C., Suci, G. J. and Tannenbaum, P. H. The measurement of meaning. Urbana, Illinois: University of Illinois Press, 1957.

Osgood, C. E. On understanding and creating sentences. American Psychologist, 1965, 18, 735-751.

Paige, J. J., and Simon, H. A. Cognitive processes in solving algebra word problems. In Kleinmuntz, B. (Ed.), Problem Solving: Research, Method and Theory. New York: John Wiley and Sons, 1966.

Petrick, S. R. A recognition procedure for transformational grammars. Unpublished Ph.D. dissertation, M.I.T., Cambridge, Massachusetts, 1965.

Piaget, J. The psychology of intelligence. Translated by M. Cook and D. E. Berlyne. London, England: Routledge and Kegan Paul, 1950.

Quillian, R. A design for an understanding machine. Paper presented at a colloquium: Semantic problems in natural language. King's College, Cambridge University, England, September, 1961.

Quillian, R. A revised design for an understanding machine. Mechanical Translation, 1962, 7, 17-29. (a)

Quillian, R. A semantic coding technique for mechanical English paraphrasing. Internal Memorandum of the Mechanical Translation Group, Research Laboratory of Electronics, M.I.T., Cambridge, Massachusetts, August, 1962. (b)

Quillian, R. A notation for representing conceptual information: an application to semantics and mechanical English paraphrasing. SP-1395, System Development Corporation, Santa Monica, California, 1963.

Quillian, R., Wortman, P. and Baylor, G. W. The programmable Piaget: behavior from the standpoint of a radical computerist. Unpublished dittoed paper, Carnegie Institute of Technology, 1965.

Raphael, B. A Computer program which "understands." Proceedings of AFIPS, 1964, Fall Joint Computer Conference, pp. 577-589.

Razran, G. H. S. A quantitative study of meaning by a conditioned salivary technique (semantic conditioning). Science, 1939a, 90, 89-90 [102, 106].

Reich, P. A. A stratificational theory of language acquisition. Working Paper #4 (IP-4), Department of Psychology and Mental Health Research Institute, University of Michigan, Ann Arbor, Michigan, 1966.

Reid, L. S., Henneman, R. H. and Long, E. R. An experimental analysis of set: the effect of categorical restriction. American Journal of Psychology, 1960, 73, 568-572.

Reiss, R. F. An abstract machine based on classical association psychology. Technical Memorandum of Librascope Division, General Precision, Inc., Glendale, California, 1961.

Reitman, W. R. Cognition and thought: an information processing approach. New York: John Wiley and Sons, Inc., 1965.

Rubenstein, H. Problems in automatic word disambiguation. Paper presented at a conference on Computer-Aided Semantic Research, held at Las Vegas, Nevada, December, 1965.

Simmons, R. F. Synthetic language behavior. Data Processing Management, 1963, 5 (12), 11-18.

Simon, H. A. and Feigenbaum, E. A. An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning. Journal of Verbal Learning and Verbal Behavior, 1964, 3, 385-397.

Simon, H. A. and Kotovsky, J. Human acquisition of concepts for sequential patterns. Psychological Review, 1963, 70, 534-546.

Skinner, B. F. Verbal Behavior. Appleton Century Crofts, Inc., New York, 1965.

Yngve, V. H. A model and an hypothesis for language structure. Proceedings of the American Philosophical Society, 1960, 104, 444-466.

Yngve, V. H. Comit Programmer's Reference Manual. Cambridge, Massachusetts: M.I.T. Press, 1961.

APPENDIX I

Input, Output & Annotated Protocol for Subject AX

A. Input, the Data Given to AX to Encode

Whip: 1. n. Stick with cord or leather fixed to end of it, used to give blows in driving animals etc or as punishment

~~2. Person helping in controlling foxhounds~~

3. Person responsible for seeing that others of his political group in Parliament are present when desired, do the right thing

1. ~~X~~. note from whip requesting person to be present in Parliament, etc.

1. ~~X~~. v.t. & i. Give blows to with whip

3. ~~X~~. Be acting as whip to (dogs, political group)

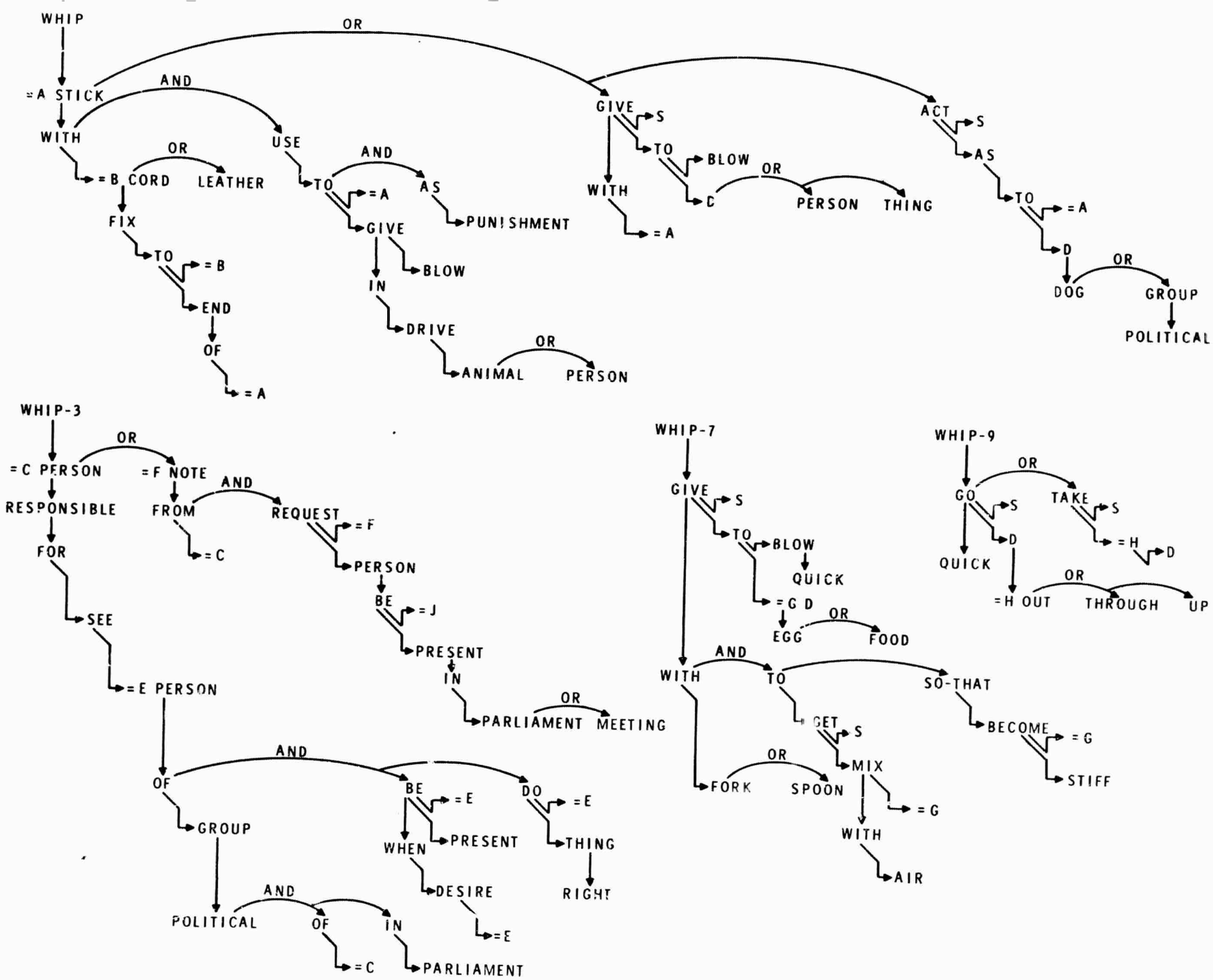
7. Give (eggs etc.) quick blows with fork etc. to get them mixed with air so that they become stiff

~~8. Get (edge of cloth) sticked with stickes which go over and through again on the other side~~

9. Go ~~in~~ or take quickly (out, through etc.)

OR

WHIP



PART B OUTPUT: THE PLANES AX CREATES TO REPRESENT THE DEFINITIONS SHOWN IN PART A.

C. Protocol and Commentary

(Note: The reader will need to refer continuously to Parts I and II, and probably sometimes to Section 1 of Chapter 2 to follow this protocol.)

	AX's Protocol		Categorization and Commentary
1	"a whip is a stick with a cord or leather fixed to the end of it used to give blows in driving animals etc. or as punishment"	1	Read. Notice that AX, instructed simply to read the definition, automatically inserts three articles and a copula which makes it into better English.
2	so the first thing for whip1, the first subclass will be a stick	2	S1 → stick. AX selects "stick" to be the 1st token in this plane.
3	which we'll make an =A because it will be used later on	3	Lb
4	since it's a noun it'll have a slash going down from it	4	Rc. Here AX has decided to use "stick" as a noun, which means it will have no subj. or obj.
		S1	"with a cord"
5	it'll be modified by "with a cord" so you do slash with	5	Seg., MB(1)
6	and then with will take cord as an object.	6	merge [prep/obj : UF(1)]
7	and then it says "with cord or leather fixed to the end of it"	7	Read.

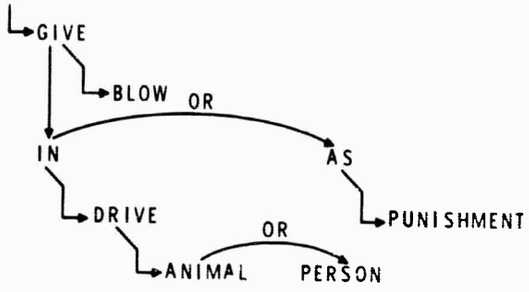
		S2	"cord or leather"
8	so "cord or leather" will will have a tag on it because I'll use it later you need a =Bcord	8	Seg., Lb
9	and then cord has an or'd set off of it, with leather	9	Rc.
		S3	"fixed to the end of it"
10	and then down from cord you have "fixed to the end of it"	10	Seg., MB(1)
11	so you do slash fix off of cord	11	SI → fix
		S4	"fixed to the end"
12	and then it's "fixed to the end" so you have the object of fix is to	12	Seg., Cf[fix/obj = UF(1)] → merge
13	and then the subject of to will be =B referring to "cord or leather"	13	Cf[to/subj:UB(1)] → merge
14	and it is "fixed to the end of the stick"	14	PI
		S5	"end of stick"
15	so the object of to is "end of stick"	15	Seg., merge [prep/obj:UF(1)]
16	end slash of	16	MB(1)
17	and then the object of of will be =A, the stick	17	merge [prep/obj:UF(1)]
18	and it says the whip is also "used to give blows in driving animals"	18	Read, MB(4)

		S6	"used to give the blows"
19	so besides being with cord, it is also "used to give the blows"	19	Seg., AX has decided she can deal with the phrase after "blows" later.
20	so off of with, you make an and'd set with use	20	Rc.
21	and then someone uses the cord to give blows	21	Rt(use/subj) → "someone" Cf[use/obj:UF(1)] → merge
22	so the object of use is to	22	merge [use/obj:UF(1)]
23	and what someone is using is the stick, =A	23	Tp → merge(use/obj:stick)
24	so =A is the subject of to	24	Rc. (AX means the object)
		S7	"give blows in driving animals"
25	and the object of to is "give blows in driving animals"	25	Seg., merge[prep/obj:UF(1)]
26	so the object of to is give	26	S1 → give
27	and what you give is blows so blows is the object of give	27	Cf[give/obj:UF(1)] → merge
		S8	"gives blows"
28	and it says "give blows" and you don't know whether you need to include an S on it, but I don't think you do because you could give one blow, so you just leave it the way it is.	28	Seg., K.
		39	"in driving animals"
29	and then you give the blows "in driving animals"	29	Seg.

30	and "in driving animals" modifies give	30	MB(2)
31	so it's a slash from give down to in	31	Rc.
		S10	"driving animals"
32	and the object of in is "driving animals"	32	Seg., merge[prep/obj:UF(1)]
33	so you leave off the 'ing' and put in drive	33	S1 → drive (as a verb, not an adjective)
34	and then what are they driving?	34	Cf(drive/obj:UF(1)) → merge
35	they're driving animals	35	
36	so animals is the object of drive	36	Rc.
		S11	"animals etc."
37	and then it says "animals etc." so you can put or person	37	Seg., K.
38	and you need to see if "drive" needs a subject and also if "give" needs a subject	38	Cf(drive/subj:?) → no match Cf(give/subj:?) → no match
39	so someone uses the stick and someone gives blows so there is no subject because whip is used as a noun here	39	Rt(use/subj) → someone; Rt(give/subj) → someone. AX also asked herself if it should be the parameter S, decided not.
		S12	"as punishment"
40	it says whip is also used "as punishment"	40	Seg.
40A	used "to give blows" or used "as punishment"	40A	merge(OR)

41 so it's used "to give blows" and "as punishment," so off of to you put an and'd set to as

41 MB(4) AX makes two errors at this point. First, it should be a disjunctive, not a conjunctive set. Second, the set should start from "in" rather than from "give." That is, the encoding structure should be:



The first error is a simple mistake, the second is due to AX's failure to note her knowledge about whips. The syntax of the text she is working on is ambiguous; apparently, she simply overlooks the possibility of attaching the phrase to "in."

42 and as will take punishment as an object

42 merge [prep/obj:UF(1)]

43	and you go on to the next use of whip1 which will be the verb use	43	Read. The fact that this use is a verb and can be transitive establishes that its plane will have an S and a D parameter. AX therefore becomes alert for a place to put these. See 46.
44	so you go back to stick and you start off an or'd set	44	Rc.
45	and the first word is — "give blows to something with the whip"	45	Seg. Notice that AX inserts an object to make English. Same as done in 47.
46	so you have — give, and give will have a subject, S	46	S1 → give, merge(give/subj:S) As explained previously, S, D, and M are parameters to be inserted into a plane by the coder whenever she knows that a sentence in which the word being coded appears ("whip" in this case) may provide them. Whenever the information "vi. and t" (=verb intransitive and transitive) preceeds a definition, AX appears to respond to it by holding in mind an S and a D, which she then inserts into that definition at appropriate places.

This is what she has done here. The point to remember as we watch AX inserting S's and D's into definitions is that where these symbols are to be placed is determined by considering various ways that "whip" can be used and not solely by thinking of how words in that definition are, or might be used.

47 someone will "give the blows to something with a whip"

47 Test

48 if you whip someone, this can be, can take, an object or not

48. Read. AX gets this information by reading "vi and t" in the definition.

49 and so, you will — the object of give will be to

49 Cf[give/obj:UF(1)] → merge

50 and the subject of to is blows

50 merge[to/subj:UB(1)]

51 and you will give it to, it can be to, D or to a person or a thing so the object is D or person and I guess you can give it to things

51 merge(prep/obj:D), K, Merge (OR)

S13 "with a whip"

52 and you do this "with a whip"

52 Seg.

53	and "with a whip" modifies give it's how you do it	53	MB(3). AX has decided that "with whip" does not modify D.
54	so it's a slash from give down to with	54	Rc.
55	and with takes whip as an object	55	merge[prep/obj:UF(1)]
56	and the whip is the =A	56	Rc.
57	and the next use is to be "acting as a whip" to in parenthesis (dogs or political group)	57	Read
58	so, with an or'd set off of stick again	58	Rc.
		S14	"to act as a whip"
59	you have act, turn the be acting into the active you make it "to act as a whip"	59	Seg., not actually a transformation, of course. AX merely simplified the language.
60	so you'll have act	60	S1 → act
61	and then the subject will be S someone will be acting	61	merge(act/subj:S)
		S15	"as a whip"
62	and how do they act? "as a whip" which is compound and they "act as a whip"	62	Seg.
63	so whip should be the object	63	merge(prep/obj:UF(1)]
		S16	"to D"
64	but they do it "to D"	64	merge(prep/obj:D), Seg. AX finds an "indirect object" and changes #63.

65	so you'll put as (as the object of act) and then you will make the object of as be to	65	Cf(act/obj:UF) → merge, merge(prep/obj:UF)
66	and then you're gonna make the subject of to be whip which is =A again	66	merge(to/subj:whip)
		S17	"dogs or political group"
67	and then the object of to will be the D which is the "dogs or political group"	67	merge(to/obj:D), Seg.
68	so you have dog and then an or'd set off of it. or group	68	merge(OR)
69	and the kind of a group is political	69	MF(1)
70	so it's a slash political and that's all for whip1	70	Rc.
71	so we go on to whip3	71	Read
72	it says the subclass is a person and the next letter is a C so you make it =Cperson	72	S1.→ person. By "the letter" AX means the next unused tag letter.
73	and what kind of a person? it's a person who is "responsible for seeing that others of his political group in parliament are present when desired or do the right thing"	73	Rc., Read. This is not coded MB(1) because there is no decision involved here. AX knows the coding system, makes the phrase modify the noun even before she reads it.

		S18	"responsible person"
74	so the kind of a person is a "responsible person," so it's a slash to <u>responsible</u> cause responsible modifies person	74	Seg., MB(1)
		S19	"responsible for seeing that others of his political group"
75	and how is he responsible? he's "responsible for seeing that others of his political group" do such and such	75	Seg., (see comment after 73) Note how AX chunks all of the phrase past "group," or at least decides that everything up to "group" can be handled correctly by so treating the rest of the sentence.
76	so it's a slash from responsible to <u>for</u>	76	MB(1)
77	and <u>for</u> will take <u>seeing</u> as an object	77	merge[prep/obj:UF(1)]
78	(Long pause) Ross, I'm afraid I'm stuck here; I don't know how to code <u>that</u>	78 and 78A	Interruption, not categorized.
78A	Experimenter: Ah..... your problem at this point was, or is, that, ah, you came to a "that." you had "person responsible for seeing that" and didn't know what to do with <u>that</u>. Well, what you should do		

78A Con't.
with the <u>that</u> is just say that the object of see is gonna be this sort of complete sentence, and so make it (the object of "see") the verb, make it <u>are present</u> or whatever the verb is and drop the that.
AX: and what about the <u>when</u>?
Experimenter: um, well, that's a <u>when</u> that means "at the time that" so you can use it.

79 whip is a "person responsible for seeing that others"

80 ah, what does he see? he's seeing this whole phrase - "Others of his political group"

79 Read.

S20 "others of his political group"

80 Seg., (here and in 81 and 82 AX delimits the object of "seeing" to the next noun phrase (rather than taking its object to be the whole "that" clause). This is a serious coding error, since the plane she now produces means: "(a whip3 is) a person responsible for seeing others of his political group in

		80	Con't. Parliament who are present when desired!" The verb must represent the clause to avoid this distortion (compare the coding of this definition by GB).
81	so this whole phrase will be the object of see	81	Cf[see/obj:UF(1)] → merge
82	the first word, instead of others we use person	82	PI
83	and to make it plural, or it can be singular, so you don't need to put a tag on it	83	K
		S21	"of his political group"
84	and what kind of a person:? they're "of his political group" which is two persons, and then	84	Seg., MB(1)
85	slash to of	85	Rc.
		S22	"political group"
86	and then "political group" is the object of of, it's group and then slash political	86	Seg., merge[prep/obj:UF(1)]
87	and it's also "of =C", you go to slash political, and then off of political is an and'd set to of	87	MF(1). She first transforms "his" to "of the person above (=C)"
		S23	"in Parliament"
88	and =C is the object of of and the group is "in Parliament"	88	Rc., Seg.

89	also off of political another and'd set "in Parliament"	89	MB(1)
90	Parliament is the object of in	90	merge[prep/obj:UF(1)]
		S2	"are present"
91	and the person is responsible for seeing that these people "are present"	91	Seg. Here AX has mentally chunked the phrase from "others" to "Parliament", and reread the sentence with this phrase represented by "these people".
92	off of of you want "are present"	92	Cf[are/subj:UB(3)] → MB(3) AX's decision here to make the verb another modifier of "person" seems clearly to be resulting from a decision that "people" is the subject of "are". A clearer coding results from putting "be" where she has "person", and then putting "person" and all its modifiers as the subject of "be". This eliminates a repetition, since she has to put "person" (=E) in as the subject of "be" anyway. (See 97 below.)
93	start an and'd set off of of	93	Rc.

94	and persons will be used again and you have to put a tag on it it'll be an =E	94	Lb.
95	and the "are present" will be be present	95	Rc. Changing "are" to "be"
96	the first word in the and'd set is be, present will be the object of be	96	Cf[are/obj:UF(1)] → merge
97	and the thing that is present are the persons, the =E (the subject of be is =E)	97	Cf[be/subj:UB(3)] → merge
		S25	"present when desired"
98	and then you have "present when desired"	98	Seg.
		S26	"when desired"
99	which, "when desired" modifies be	99	MB(1). Note that AX also indicates that she is thinking of "be present" as a chunked segment, since she left out "be" in 97.
100	so that's a slash to when from be	100	Rc.
101	and then when takes an object which is desired	101	merge[prep/obj:UF(1)]
102	and desired is passive so you have to put whatever is desired, which is the person, is =E	102	Tp → Cf[desire/obj:UB(4) → merge. An error; the object of "desire" should be "be present."
103	(desire) will take an object	103	Rc.

104	off of desire you'll put an =E as the object	104	Rc.
		S27	"do the right thing"
105	besides seeing that the people are present he must see if they "do the right thing"	105	Seg. Cf[do/subj:UB(5)] → merge
106	so off of of make another and'd set besides be (so you have an and'd set to do)	106	MB(5), AX really means extend the conjunctive set.
107	put "do the right thing"	107	Rc.
108	this S(ubject) will also be =E, =E is to do the right thing	108	merge[do/subj:MB(5)]. Test.
109	=E is the subject of do	109	Rc.
110	and the object of do is thing slash right	110	Cf[do/obj:UF(1)] → me.~e
110A	(and the kind of thing is right)	110A	MB(1), Rc.
111	and the next use for whip3 is a "note from a whip requesting a person to be present in Parliament"	111	Read.
112	off of =Cperson you start an or'd set	112	Rc.
113	the first word will be a note	113	S1. → note
114	and you will need a tag on that which will be =F	114	Lb.
		S28	"from a whip"
115	what kind of a note? it's "from a whip"	115	Seg., MB(1) AX has also decided "note" is a noun.

116	so you put a slash from	116	Rc.
117	and the object of from will be whip which is the person responsible which is the =C	117	Cf[prep/obj:UF(1)] → merge
		S29	"requests a person to be present"
118	and the note besides being "from a whip" it also "requests a person to be present"	118	Seg., Cf[requests/subj:UB (2)] → MB(2), K. We call this K because AX's decision here that "request" applies to the "note" rather than to the "whip" seems dependent on her knowledge of what this kind of "whip" is.
119	so off of from will be an and'd set starting with request	119	S1. → request, Rc.
120	and what requests? the subject of request is the note, =F	120	merge[request/subj:UB(2)]
		S30	"for a person to be present"
121	and what it requests is "for a person to be present"	121	Seg., Cf[request/obj:UB(1)] → merg
122	and, I guess it's person is the object of request, or to could be the object um the note, the =F, requests that the "person be present in Parliament"	122	S1. → "person" or "to?" Test. AX rereads whole phrase, decides to omit "to," seems to be testing if this reads all right.
123	therefore the note is modified by request	123	Test.

124	request will take person as an object	124	S1. → person
125	person will be become an =J	125	Lb.
		S31	"is present in Parliament"
126	and then person will have a slash to be, the kind of person is the person that "is present in Parliament"	126	Cf[be/subj:UB(2)] → MB(2). Better coding again, would be to put "be" where AX puts "person."
127	so the subject of be will be =J, person	127	merge [be/subj:UB(2)]
128	and the object of be will be present	128	Cf[be/obj:UF(1)] → merge. AX also decides that "present" is a noun here.
		S32	"present in Parliament"
129	and the kind of present it is is "present in Parliament"	129	Seg.
		S33	"in Parliament"
130	so present is modified by "in Parliament"	130	Seg., MB(1)
131	so it's slash in	131	Rc.
132	and the object of in is Parliament	132	merge[prep/obj:UF(1)]
		S34	"or meeting"
132A	or etc., which can be "or meeting" (start an or'd set off of Parliament)	132A	Seg., K.
133	then, whip7 is an or'd set off of whip3	133	Read.

134	and this is "to give eggs quick blows with fork etc. to get them mixed with air so that they become stiff"	134	Read: notice that AX omits the first "etc.," in attempting to read the phrase verbatim. This indicates that she has already chunked "etc." with "eggs."
135	so the first word is <u>give</u>	135	S1. → give
136	give will have a subject which is <u>S</u>	136	merge(give/subj:S).
137	and the object is <u>eggs</u> and what he is giving is blows	137	Cf[give/obj:UF(1)] → merge, but also, Cf[give/obj:UF(2)] → merge
		S35	"giving to the eggs" "quick blows"
138	so he is "giving to the eggs," I guess, "quick blows"	138	Seg., at this point, AX generates the word "to" to permit her to represent the two (direct and indirect) objects
139	so, <u>blows</u> can be the object of give or the object of eggs um I'll try it	139	Test. In this interesting statement AX considers using "eggs" as a verb or preposition, but presumably rejects this when she rereads the sentence to herself. Considered in terms of semantic context AX's idea may be less far-fetched than it at first sounds, since the verb "to egg someone on," is very often used in a context concerned with "blows."

140	I'll think I'll make eggs the object of blows	140	Cf[blows/obj:UB(1)] → ?. Here she considers using "blows" as a verb.
141	and make blows, could make it "give quick blows to the eggs" yeah I think I'll do that	141	Test → reject. Test. Here she apparently reads off the structure implied by 140 and perceives its error. She tries to correct this by inserting the word "to," tests the phrase this produces, and finds it correct.
142	so to will be the object of give and then you'll put to D	142	Cf(give/obj:to) → merge, inserts D.
		S36	"or food"
143	and the object of to will be D slash egg, and then there's an etc., so you put "or food" (start an and'd set off of egg)	143	Cf(prep/obj:D) → merge, Seg.
144	and the subject of to will be "quick blows" (so blow is the subject of to)	144	Cf[to/subj:UF(1)(unit following the verb)] → merge.
145	so it's blow slash quick	145	MF(1)
146	and there can be one or more blows so you don't pay any attention to the S	146	K.
		S37	"with a fork etc."
146A	and you do this "with a fork etc." so this modifies give, and it's give slash with	146A	Seg., MB(3). AX decides that the chunk "with a fork" modifies the verb, rather than "blows" or "eggs, etc."

		S38	"fork etc."
146B	a "fork etc." is the object of with	146B	Seg., merge[prep/obj:UF(1)]
		S39	"or spoon"
146C	etc. can be an or'd set off of fork to "or spoon."	146C	Seg., K.
		S40	"get the eggs mixed with air"
147	and you do this to "get the eggs mixed with air"	147	Seg., PI
		S41	"to get them mixed"
148	"to get them mixed modifies give	148	Seg., MB(4)
149	so from give, you also have to and to will have get as an object and the subject is getting, the subject of get is S	149	Rc., merge[prep/obj:UF(1)] → merge Test → merge (get/subj:S)
150	and he is getting the eggs mixed, or getting the D mixed, with air	150	Test. AX decides the eggs are D.
151	since you'll have to use the D again, I'll have to put a tag on it so I'll make it an =G, =GD slash eggs or food	151	Lb.
152	so the subject is "getting the eggs to be mixed with air"	152	Test → reject. (Here AX erased the structure in which "eggs" was the object of "get.")
153	or you can have the "subject gets mixed"	153	Test.

154	and make mix the object of get	154	Cf[get/obj:UF(1)] → merge. Here AX codes "correctly" making the verb represent the phrase.
155	so getting mixed	155	Test.
156	what is getting mixed? so the object of mix will be the eggs, which is the =G	156	Cf[mix/obj:UB(1)] → merge. Here, UB(1) is not a very good characterization of "them;" actually it is in the middle of the compound verb.
		S42	"mixed with air"
157	and you want them "mixed with air"	157	Seg.
		S43	"with air"
158	so "with air" can modify mix, or it can modify get, you can have "get with air," that doesn't make sense	158	Seg., MB(1) or MB(2)? → Test → reject MB(2). This is a clear example of AX feeding two possible segments to her non-conscious language processor and getting its results
159	so it's "get mixed with air"	159	Test.
160	so "with air" will modify mix (so it's slash with from mix and air is the object of with)	160	Rc.(158), Cf[mix/obj:UF(1)] → merge.
		S44	"so that they become stiff"
161	and you do this "so that they become stiff"	161	Seg. AX has already decided that this phrase will modify the action somehow.

		S45	"so that"
162	so that is a problem, so you give blows "so that"...	162	Seg., Test
163	well, you can put it off of the to get, cause it sort of modifies give but, I guess I'll do that	163	Text., MB(3), we count this MB(3) rather than MB(7) because we assume that only verbs are acceptable things for "so that" to modify. Note that in the test sentence AX uses here she includes the object ("blows") but omits its modifiers, etc.
164	you can make a compound out of it	164	K. AX means to put a hyphen into "so-that" and treat it as a single node.
165	yeah, so you'll put an and'd set off of the to get	165	Rc.
166	and have it so-that make a compound out of it	166	Rc.
		S46	"the eggs become stiff"
167	and then it will take an object which is so that "the eggs become stiff"	167	Seg., Cf[so-that/obj:UF(1)] → merge Here the UF is the whole phrase.
168	so the object of so-that is become	168	S1. = become.
169	and the subject of become is is the eggs again which is the =G	169	PI, Cf[become/subj:UB(1)] → merge
170	the object of become is stiff	170	Cf[become/obj:UF(1)] → merge
171	whip9 which is an or'd off of whip1	171	Read.

172	you do a slash	172	Rc.
173	and this is also a verb	173	Read.
174	and it's to "go or take quickly out or through"	174	Read. This is a mistake; what the dictionary definition really means here is: "go or take quickly, as for instance in 'whip out,' 'whip through,' etc." AX however, continues to think that the parenthesized words again describe the D, and a rather confused and needlessly complicated coding results.
		S47	"goes quickly out or through something"
175	so it would mean that a person "goes quickly out or through something"	175	Read. In this rereading AX apparently segments "go or take," and supplies a subject and object to increase its understandability.
176	to whip something up is to go quickly through it, to get it done fast	176	K., She seems to be trying to relate the definition to her prior knowledge.
177	so you have (slash) go	177	S1. → go
178	and then you have or (take) off of that (an or'd set off of go to take)	178	Rc.
179	and quickly modifies both of them	179	MB(1)
180	so it's a slash from go for quick	180	Rc.
181	and the subject of go is S	181	merge(go/subj:S)

		S48	"out or through"
182	the subject goes "out or through" something--he whips through something	182	Seg.
183	so the "out or through the thing" is the D, I think	183	Read. Here and below AX is struggling to relate this definition to her prior knowledge. Notice how her speech generator insists on an object for the prepositions "out or through," and inserts its own ("the thing") when none is given.
184	he whips the thing, he whips through the thing	184	Test.
185	so if the subject whips through something...	185	Test, more struggling to understand the definition.
186	so the D will be "out or through"	186	Cf[whip/obj.UF(1)(unit following "go or take")] → merge.
187	so you "go or take quickly out or through"	187	Test.
		S49	"out or through or up"
188	"out or through or up" is the D for go but in take, you can take something, D, "quickly out or through"	188	Seg., Test, Cf[take/obj:UF(1)] → merge. Here the UF(1) is D, but AX realizes that "out or through" is part of the verb, which puts D in the position of an indirect object
189	I think, so the subject can take an object, "out or through"	189	Test.

190	so I'll put a tag on "out or through" as an =H	190	Lb.
191	so the object of take will be =H	191	merge[take/obj:UF(1)]
192	and the subject of =H will be D	192	merge("out or through"/ subj:D)
193	I'm not sure, but I think that's right	193	AX has trouble understanding this last definition. In the differing constructions beneath go and take she has attempted to account for the fact that the object of whip9 can be either a prepositional phrase, as in "whip through the work," or a noun phrase, as in "the driver whipped the car around the curve."

D. Categories Used to Classify the Steps Taken by Subject AX in the Protocol

Term or Category	Explanation
UF(1)	This term is used to designate the Unit Forward by one (the next unit) in the text. What the unit is will always be clear from the context; it can be a noun phrase, a prepositional phrase, or a whole clause. UF(2) means the Unit Forward by two, i.e., with one unit interposed between it and the word or unit being taken as a reference point.
UB(1)	This term means the Unit Backward one, i.e., the immediately preceding unit in the text. UB(2) is the unit back two, and so on.
Merge[prep/obj:UF(1)]	This category would be used to characterize a step in the protocol at which AX asserts that the object of some preposition is the Unit Forward by one in the text.
Cf[go/obj:UF(1)]	This category would appear if AX were to compare what she has stored in her memory as a likely object for the word "go" to the unit just forward of "go" in the text. The abbreviation "obj" would be replaced by "subj" if she were considering UF(1) as a subject for the word "go." In place of "go" any word can appear, and in place of UF(1) any symbol such as UB(1), UF(3), etc., may appear. The Cf category is usually followed by a symbol → merge. This means that as a result of AX's

comparing some unit to the subj or obj "slot," she has decided to go ahead and merge them. Thus, at #27 in the protocol, our terminology: Cf(give/obj:UF(1)) → merge, means that AX has apparently compared the word immediately following "give" in the text, "blows," to what she has stored as a likely object for "give," found the match acceptable, and decided to make it. The Cf category clearly involves an inference by the categorizer as to AX's information processing; all other categories are more directly descriptive.

Seg. — This is used when AX segments off a continuous segment of the text for consideration. To be categorized Seg., a paragraph of the protocol must contain the first explicit reading aloud of some delimited segment of text, and moreover, have been put into quotes by AX when she edited the protocol. Long stretches of the text, such as those including all of one definition, are not categorized as Seg. A stretch of text isolated by Seg. is then typically a unit, and may constitute a UB later in the analysis.

Rt. — AX is retrieving a word from memory.

MB(n) where n is a small integer. — This is used wherever AX decides that the current unit she is working on is to Modify some other unit that is n units Back in the text.

MF(n), where n is a small integer — Used wherever AX decides that some unit is to Modify another unit n units Forward of it in the text.

S1 → hay	This category means AX is selecting some word (in this case, "hay") to represent some definition or segment of a definition previously isolated. By a word's representing a segment we mean that it will appear in the plane above any of the other words in that segment.
PI	Pronoun identification. Used wherever AX identifies the referent of some pronoun.
Rc.	AX is Recording the result of some prior decision by writing it onto the plane.
Lb.	AX labels some token node of the plane as =A, =B, etc., in order to be able to refer to it later on.
Test	AX tests some potential or recorded piece of encoding by reading it aloud in its English equivalent.
Tp.	AX transforms a passive sentence or phrase into active form.
K	AX resorts to prior knowledge in order to make some decision. An example is when she replaces the word "etc." with some word specifying what "etc." means in that particular case.
Read	AX reads aloud a definition.

In Part III, AX's protocol (with subheadings numbered S1, S2, S3, etc., inserted to show the segment she is working on), is shown on the left side of the page. The categorization of each step of the protocol into one or more of the preceding twelve categories and our commentary are shown on the right.

APPENDIX II

Number of Computer Parsings for Seven Sentences Analyzed by the Harvard Multiple-Path Syntactic Analyzer[1]

Number of Parsings	Sentence
120	1. A whip can be a stick with cord or leather fixed to the end of it, used to give blows in driving animals etc., or as punishment.
24	2. A whip can be a person responsible for seeing that others of his political group in parliament are present when desired, or do the right thing.
1	3. A whip can be a note from a whip requesting a person to be present in parliament, etc.
1	4. To whip can be to give blows to something with a whip.
0	5. To whip can be to be acting as a whip to dogs or a political group.
1066	6. To whip can be to give eggs, etc., quick blows with fork, etc., to get them mixed with air so that they become stiff.
28	7. To whip can be to go or take something quickly out or through.

[1] See Kuno, 1963, 1965.

Note: The numbers above include only parsings actually enumerated by the program. In these, the program does not bother to enumerate all the ways that prepositional phrases and present participle clauses might modify nouns and verbs of the sentence.

Instead, it simply shows the parsings which assign these to their immediately preceding noun or verb. (We will argue in Chapter VII that these assignments cannot in general be made without a sophisticated use of semantics.) The importance of such assignments may be seen in sentence 3, where the parsing that the program explicitly produced incorrectly makes "requesting a person" modify "whip" rather than "note." In order to always include the sentence parsing which has the correct assignment of all such phrases, one must consider all combinations of possible phrase assignments, which can increase the number of potential parsings by a sizeable factor — e.g., for sentence 1, by a factor of 540.

Also, the program's parsings fail to include the correct parsing for sentences 1 and 4, because it had "blows" coded only as a verb, whereas in these sentences it is a noun. This difficulty is trivial to correct, but, again, would multiply the number of parsings of these sentences by some unknown factor.

By way of illustration, we isolate below the ambiguities, combinations of which produced the 24 parsings of sentence 2.

WORD	AMBIGUITY
do	1) start of a declarative clause. 2) start of an imperative (!) clause.
that	1) introducing object clause (as in "seeing that something is true") 2) describing the prior verb (as in "seeing, which is good") 3) as a pro-adjective (as in "Did you see that ball?)"
desired	1) as a gerund 2) as an object clause
right	1) as a noun 2) as an adjective

APPENDIX III

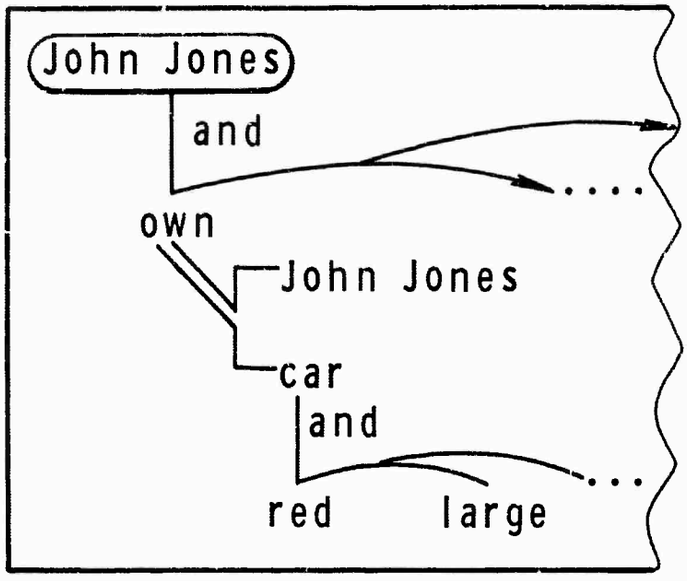
Recognizing Synonymous Statements in the Memory

Here an example is presented to illustrate how a reader may be able to locate in his memory associations equivalent to those made in a text, even though their association is stored in different terms from that the text employs. Our example will illustrate why it was stated in Chapter II that the parameter symbols, S, D, and M, are essential to being able to recognize the synonymy of statements.

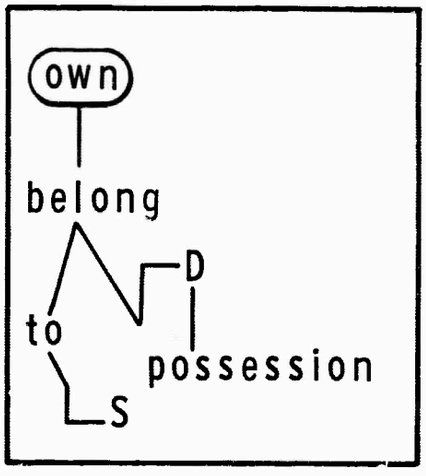
Note first that a proper name, like John Jones, can be the type node heading a plane of information, just as some single word can be. Suppose that some reader has a plane in his memory for John Jones, and that this contains token nodes which express a large amount of information he knows about John Jones, including the fact that John Jones owns a large red car. This plane is indicated in Figure 7A.

Now, this reader attempts to read part of a text which states, "....and the red car belongs to John." The section of a plane this corresponds to is shown as Figure 7C. According to our theory of text understanding, this reader will attempt to discover the place in his memory at which he has already recorded this fact. Since, let us say, he knows several people named John, he will have to look at the plane of each of these to see if any contains the present text's information. If the right one can be found, it will allow the reader to decide that

7A. A Hypothetical John Jones Plane



7B. A Hypothetical Plane Defining "Own"



7C. Section Of A Plane Representing:
"....the red car belongs to John."

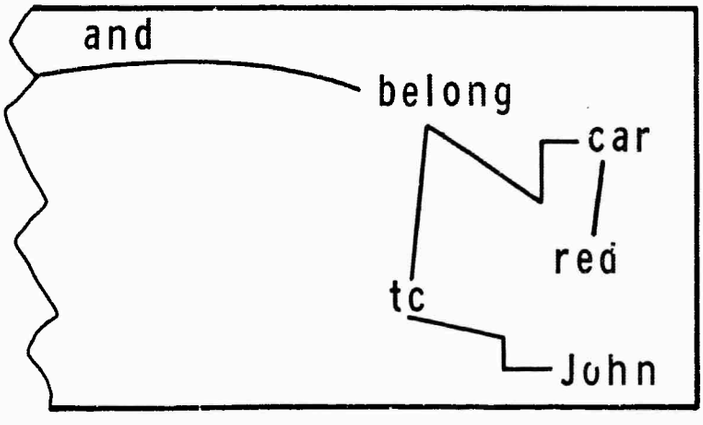


FIG. 7 THREE HYPOTHETICAL PLANES

it is John Jones that the text is probably referring to, and consequently to apply all the other knowledge stored with the John Jones plane in interpreting and evaluating future parts of the text. The question, then, is how might the reader realize that his memory contains a statement equivalent to what the text is now stating in different terms?

To take the simplest case first, let us suppose that his memory also contains a plane defining the word "own" which looks like that of Figure 7B. This plane expresses the definition that the word "own" can refer to the state in which "(some possession) belongs to someone." Now, suppose the reader tentatively uses the "John Jones" node and the "belong" node — provided by the text — as two patriarchs to be intersected by a program like that of Chapter III. This should find a path going through the token node for "own" in his plane for John Jones, and then through its token node for "belong" to the type node for "belong." Finding this path could be accomplished by the program of Chapter III.

However, in order to recognize the synonymy of the statement stored in memory to that in the text, it is also necessary that the reader's text processor know how to replace the parameter symbols S and D in the plane defining "own," w th "John Jones" and "red car," respectively. The placement of the parameter symbol S in Figure 7B specifies that whatever is related to any token of "own" as its subject (_, see key to Figure 1, Chapter II) is to be temporarily "placed" into its defining plane as the object (_) of the node "to." Correspondingly, the symbol D in the "own" plane specifies that whatever was related to a token for "own" as an object is to be "placed" into its defining plane as the subject of "belong." Once this is done, this plane in his memory is in a comparable

form to the plane (Figure 7C) representing the statement in the text, and can be seen to subsume it.

Thus the parameter symbols, S, D, and M, not only say how subjects, objects and words modified by a word, when it is used in text, are to be placed into its defining plane -- their original purpose, see Section C of Chapter II -- but also say how nodes connected to one of its token nodes elsewhere in memory are to be "carried across" in order to recast a fact stated in one form into a synonymous form.

The above was almost the simplest possible case. In a slightly more complex case, the reader's definition plane for "own" would not happen to contain a token for "belong." However, in this case it might well contain some other word, say "X," whose definition does contain "belong." In this case, the S's, D's and M's in the definition of the word "X" will tell how to carry across "John Jones" and "red car" into the plane of word "X," and then the S's, D's and M's of the "belong" plane will again tell how to carry across information from the token for "belong" from the plane defining "X." That is the second kind of case. In yet a worse case, in order to intersect "John Jones" and "belong," paths would have to move out from both of these patriarchs to some intersection node. (e.g., see Figure 2B in Chapter III). In this case, "John Jones" and "red car" would have to be "carried" both from the John Jones plane in memory, and from the plane established by the text, in order to investigate synonymy (or contradiction) between the assertion stored in memory and that of the text. Without question, unforeseen problems lie in the way of making a computer implementation of such "carrying across" really work; this example is included only to show that recognizing identity of meaning between differently worded statements is

theoretically feasible in a memory of this sort we have been discussing, and to explain why the investigation of parameter symbols such as S, D, and M is of primary importance in future research.[1]

---

[1]See Chapter VIII for why S, D, and M, are probable too crudely defined.

APPENDIX IV

TURNING PATHS INTO ENGLISH TEXT

The sentence production program which actually produced the English output of Table 1 consists of approximately 2,000 IPL instructions. Since that program was built up more by trial and error than as the realization of any comprehensive plan, and since in it various kinds of trivial housekeeping functions are intertwined with logically significant operations, the effort necessary to understand it is great, and relatively unrewarding. For the reader who does want to delve more deeply into just how such a program can operate, we believe the set of rules presented below will be useful.

These rules constitute the crucial part of such a sentence production program. They were actually written for a later program, one designed to work like the one described in Chapter III, except that in this later program the sentence production rules are isolated from their interpretive and housekeeping mechanisms. Besides being more intelligible, it is hoped that this later program will enable experimenting with changes in the sentence production rules without changing any other component of the system, and hence will enable us to experiment with expressing the same path in various language styles.

The set of rules below have come to be called the "Hemmingway Grammar," because of their predilection for the simplest, often highly repetitious way of expressing anything.

Other such sets of rules (culminating, eventually, in a triumphant "Henry James Grammar"?) are contemplated.

Since the "Hemmingway Grammar" has never actually been run, it undoubtedly contains bugs and logical errors. However, it does at least enumerate a number of the kinds of path segments that will be encountered by such a program, and provides a first attempt at a way of expressing each of these. Thus, study of it should reveal the interesting aspects of such a program's operation, even though it cannot be taken as more than a first approximation to a running program.

Since these rules are designed to operate on paths produced by a LISP program rather than on paths from the IPL program, it will be necessary for the reader to learn the new types of connectives utilized in this LISP memory model before he can follow the grammar. The most important change made in this LISP model is that all prepositions are now used as connectives. (See Part A of Chapter VIII). A path given to this set of rules is always a list, alternating between substantive nodes and connectives, as:

substantive node + connective + substantive node
+ connective + substantive node + ... +
connective + substantive node

A "substantive node" can be a single substantive word, or itself a list starting with a word (or with "AND," "OR," "=," or "LB," see below) and followed, again, by a string in which connectives and substantive nodes alternate. Thus a path is a list structure with embedded elements, but in which embedding is restricted to alternate members of each string. By way of example, the LISP path that would correspond most

closely to the lefthand path shown in Figure 2-b is: cry DEF OR 2 cry2 TIS cry2 /c AND 2 make OB sound5 / sad TIS sad (parentheses have been removed and connectives underlined for easier readability.)

The legal connectives in the LISP model are as follows:

Table of Connectives in the LISP Memory Model

Connective	Interpretation	Closest Equivalent in IPL Program (See key to Figure 1)
TIS	Current node is token, for which next node is the corresponding type.	Link 6
/	Next node is (simple adjectival or adverbial) delimiter of the current node	Link 2
DEF	Current node is defined by the next (complex) node.	Link 1
SB	Next node is the subject of the current node.	Link 5 (upper line)
OB	Next node is object of current node.	Link 5 (lower line)
/S	Next node will have an SB, which will point to another token for the current node.	Link 2
/O	Next node will have an OB, which will point to another token for the current node.	Link 2

Connective	Interpretation	Closest Equivalent in IPL Program (See key to Figure 1)
/C	Next node is an alternative way of naming the same concept the current node names. Note that this is not the same as naming some one of a set of more explicit concepts which the current node may be used to name, in which case a disjunctive set must be formed.	Link 2
PREP (for example, "by," by5, to3, etc.	Current node is related to next node as this preposition specifies.	Link 2

Note that OR and AND, link types 3 and 4 in the IPL program, are treated in the LISP structure not as connectives, but as the names of special nodes. (These are removed from a path by the pre-editing rules at the start of the sentence production rules. The pre-editing rules also remove the symbols that indicate back pointers ("=," and "LB") from a path.)

The sentence production rules are to be interpreted by a more general interpretive program which operates as follows: NEXT is the name of a variable, whose value is a location in the path that is being worked on. When the program first begins to operate on a path, NEXT is initialized to the first element in that path; as various rules operate, NEXT is advanced progressively along the path. The interpreter proceeds

through the rules in order, attempting to match the elements of the path which surround the NEXT item with the left side of each rule. When this match fails, the program goes on to the following rule. When this match succeeds, the matched parts of the path are rewritten as specified in the right side of that same rule. In the right side of each rule, one item appears underlined, to indicate which item the variable NEXT is to be reset to whenever that rule operates. After a rule is effected and NEXT has been reset, the interpreter proceeds to the rule specified after the symbol GO. (The reader unfamiliar with such string manipulation interpreters may see, for example, Yngve, 1961, or Bobrow and Teitelman, 1966.)

In the rules the symbol "$1" refers to any one node. The symbol "NIL" stands for nothing, i.e., a path that is ended. The symbol "V" stands for "or." Parentheses in the rules or examples refer to optional or alternative items. The symbol "Verb[NEXT]" means to test if the node pointed to by NEXT is a verb. The symbol "SB[X]" means to get the subject of X.

It is also understood that a set of "morphophonemic" rules will operate on the path after the sentence production rules have completed their processing of it. The morphophonemic component will add all adjectives, adverbs, and objects not already in the path, and make other slight alterations. Thus, the example path above would be rewritten by the grammar to form:

cry be cry2 . ANAP cry2 be make sound5 . ANAP sound5 be sad . ANAP

Then the morphophonemic component should take this to:

To cry can be to cry2. This crying can be to make a sound. This sound is sad.

Other examples of the sorts of actions the morphophonemic rules must perform are:

Noun[$1] → (a, an, no, not a, not an, nil) + (adjectives) + 1 + (prepositional phrases)

(Parenthesized elements are added if these are available on a token but are not otherwise mentioned in the path)

Verb[$1] → (to, not to, not) + 1 + (objects) + (adverbials)

Be → is, are, etc.

PRON/X → he, she, it, they, him, her, itself, himself, herself, themselves. X is the referent of the pronoun, if that referent hasn't previously appeared in the sentence PRON/X is rendered as X.

POSS-PRON/X → his, her, their, its, etc.

ANAP + NIL → nil

ANAP + Verb → this + 2ing. Otherwise ANAP → (this, these, the)

DEM/X → which, who

Finally, it is essential to bear in mind that the nodes that appear in a path actually name nodes in the memory model, which is not a linear but a branching structure. Since a path does not branch, it in effect selects one branch out of all those available at each node. The sentence producer can and often does look at the other branches from a node, although at the same time it must keep track of just which link is mentioned in the path, since it is this aspect of the stored information that it is currently necessary to express (and, if the path continues, to elaborate further upon.)

Tentative Sentence Production Rules ("Hemmingway Grammar")

Note: The examples illustrated under each rule show the segment of a text which might come out of such a rule as they would appear after the operation of the morphophonemic rules.

Rule Number	Left Side	Right Side		Mnemonic Rule Name
2.	LB + $1	→ nil	GO 2	
3.	= + $1 + $1	→ pron/3	GO 3	
4.	TIS + $1	→ nil	GO 4	pre-editing
5.	AND + number[$1] + $1	→ 3/&	GO 5	
6.	OR + number[$1] + $1	→ 3/V	GO 6	

(At this point the variable NEXT is initialized to point to the first item in path.)

10.	NEXT + nil	→ nil	GO morpho-phonemics	End of text
30.	NEXT + (DEF / /c) + $1 "To chase is to pursue" "To hurry is to move quickly" "A car is a vehicle"	→ 1 + be + 3	GO 100	copulative
31.	NEXT + Prep[$1] + $1 "A cockpit is in a plane. A plane...."	→ 1 + be + 2 + 3	GO 100	Copulative prepositional
35.	NEXT + SB + $1 "This flying is done by a pilot. A pilot..."	→ 1/ing + be + done + by + 3 + . + 3	GO 10	Subject Elaboration

Rule Number	Left Side	Right Side	Mnemonic Rule Name
36.	NEXT + OB + $1	→ 3 + be + 1/en + . + 3 GO 10	Object Elaboration
	"Food is eaten. Food...."		
40.	NEXT + /O + $1 + SB + $1	→ OB[3] + be + 3/ed + by + 5 GO 100	Passive Subject Elaboration
	"A car is used by a man. This man..."		
49.	NEXT + /S + $1 + OB + $1 + DEF	→ SB[3] + 3 + 5 + , + dem/5 + 6 GO 10	Active Object Definition
	"This man drives a car, which is..."		
50.	NEXT + /s + $1 + OB + $1	→ SB[3] + 3 + 5 + that GO 10	Active Object Elaboration
	"This man drives a car that...."		
71.	$1 + NEXT + /O + $1 + OB + $1	→ IF + 1 + OB[4] + be + 4/ed + PRON/2 GO 10	Passive Object Elaboration
	"If this (brown) car is used, it....(verb, copula)		
	or		
		→ 1 + OB[4] + be + 4/ed + , + AND + PRON/1 GO 10	
	"This car is used, and it...(verb, copula)"		
73.	NEXT + /S + $1 + SB + $1	→ SB[3] + 3 + (OB[3]) + AND + PRON/1 GO 10	Active Subject Elaboration
	"This car uses (oil) and it -----" "This car runs and it -----"		

Rule Number	Left Side	Right Side		Mnemonic Rule Name
74.	NEXT + /S + $1 + DEF + $1	→ SB[3] + 3 + (OB[3]) + . + To + do + so + be + 5	GO 100	Verb Elaboration
	"This man runs a gas station. To do so is...."			
75.	NEXT + /S + $1 + / + $1	→ SB[3] + 3 + 5 + . + 5	GO 10	Verb + Adverb
	"This man moves slowly. Slowly...."			
76.	NEXT + /S + $1 + /c	→ SB[3] + 3 + (OB[3]) + . + POSS PRON/1 + 3/ing + 4	GO 10	Active Clause Elaboration
	"This man drives a car. His driving..."			
77.	NEXT + /S + $1 + Prep [$1] + $1	→ SB[3] + 3 + 4 + 5	GO 100	Prepositional Object Elaboration
	"This man runs near a lake. This lake..."			
79.	NEXT + /O + $1 + DEF	→ OB [3] + be + 3/ed + . + 3 + 4	GO 10	Passive Verb Elaboration
	"This man is frightened (easily). To frighten...."			
80.	NEXT + /O + $1 + / + $1	→ OB [3] + be + 3/ed + 5 + ly + . + 5	GO 10	Passive Verb and Adverb
	"This man is frightened easily. Easily...."			
81.	NEXT + /O + $1 + /c	→ OB [3] + be + 3/ed + . + POSS. Pron/1 + 3/ing + 4	GO 10	Passive Clause Elaboration
	"This man is frightened (easily). His frightening...."			

Rule Number	Left Side	Right Side		Mnemonic Rule Name
82.	NEXT + /O + $1 + Prep [$1] + $1	→ OB [3] + be + 3/ed + 4 + 5	GO 100	Passive Prep Phrase
	"This man is frightened in the dark. This dark...."			
83.	NEXT + /O + $1 + NIL	→ OB [3] + be + 3/ed + .	GO MORPHO	Passive End of Text
	"This man is respected (widely)."			
84.	NEXT + / S + $1 + NIL	→ SB [3] + 3 + .	GO MORPHO	Active End of Text
	"This man runs (a gas station) (well).."			
90.	NEXT	→ Unforeseen Pattern		Error Routine Print out
100.	VERB [NEXT] + / + $1	→ 1 + 3/ly + . + 3	GO 10	End Sentence with Adverb
101.	VERB [NEXT] + OB + $1	→ 1 + 3	GO 100	End Sentence with object(s)
102.	PRON.[NEXT]	→ 1/ACC + . + 1	GO 10	End Sentence with Pronoun
103.	NEXT	→ 1 + . + ANAP + 1	GO 10	End Other Sentence

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
SEMANTIC MEMORY

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific Report No. 2

5. AUTHOR(S) *(Last name, first name, initial)*
Quillian, M. Ross

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1966	231	78

8a. CONTRACT OR GRANT NO.
AF19(628)-5065 ARPA No. 627 Amendment No.2

b. PROJECT AND TASK NO.
8668

c. DOD ELEMENT

d. DOD SUBELEMENT

9a. ORIGINATOR'S REPORT NUMBER(S)
BBN Report #1352

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
AFCRL-66-189

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES
The work reported was supported by the Advanced Res. Projects Agency, dtd March 1965

12. SPONSORING MILITARY ACTIVITY
Hq. AFCRL, OAR (CRB)
USAF, L G Hanscom Field
Bedford, Mass. 01730

13. ABSTRACT

This report describes a model for the general structure of human long term memory. In this model, information about such things as the meanings of words is stored in a complex network, which then displays some of the desirable properties of a human's semantic memory. Most important of these properties is the capability of the memory to be used inferentially; i.e., to allow for the answering of questions besides those specifically anticipated at the time the information is stored in the memory. A computer program is described which illustrates this property by using the memory model inferentially to simulate human performance on a basic semantic task.

When the meaning of some segment of natural language text is represented in the format of the model, relationships and features of this meaning must be made explicit which were not explicit in the text itself. This becomes a methodological advantage in an experiment in which a person interprets text, as described.

DD FORM 1 JAN 64 1473

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
semantic memory models						
semantic analysis						
linguistic performance models						
natural language						
human memory						
human understanding of text						
simulation						
syntactic analysis						
semantic networks						
linguistics						
computational linguistics						
list processing						
information retrieval						
protocol analysis						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.