# GENERATING ENGLISH DISCOURSE

# FROM SEMANTIC NETWORKS

*TECHNICAL REPORT NO. NL-3*

*R. F. Simmons*
*Jonathan Slocum*

*November 1970*

*NATURAL LANGUAGE RESEARCH FOR COMPUTER-ASSISTED INSTRUCTION*

*Supported By:*

*THE NATIONAL SCIENCE FOUNDATION*
*Grant GJ 509 X*

*Department of Computer Sciences*

*and*

*Computer-Assisted Instruction Laboratory*

*The University of Texas*
*Austin, Texas*

# GENERATING ENGLISH DISCOURSE FROM SEMANTIC NETWORKS

R. Simmons & J. Slocum

## Abstract

A system is described for generating English sentences from a form of semantic nets in which the nodes are word-sense meanings and the paths are primarily deep case relations. The grammar used by the system is in the form of a network that imposes an ordering on a set of syntactic transformations that are expressed as LISP functions. The generation algorithm uses the information in the semantic network to select appropriate generation paths through the grammar.

The system is designed for use as a computational tool that allows a linguist to develop and study methods for generating surface strings from an underlying semantic structure. Initial findings with regard to form determiners such as Voice, Form, Tense, and Mood, some rules for embedding sentences, and some attention to pronominal substitution are reported. The system is programmed in LISP 1.5 and is available from the authors.

# GENERATING ENGLISH DISCOURSE FROM SEMANTIC NETWORKS *

R. Simmons & J. Slocum

## I Introduction

Much of the recent work in language processing research has been concerned with representing factual material in the form of semantic nets for the purpose of answering questions, guiding students in computer aided instruction and counseling, solving problems, etc. In a previous paper a detailed definition of semantic network representations for aspects of English meanings was developed (Simmons 1970b). That paper presented methods for representing the semantic structure of English discourse and lexical information as a network of word-sense concepts connected by deep case relations. Here, after a brief discussion of its place in linguistic theory, an algorithm and a grammar will be described to generate coherent sequences of sentences from the semantic network and its associated lexicon.

Background: Recent variations of transformational theories of linguistics proposed by Lakoff (1969) and Lakoff & Ross (1969) suggest that the process of generating natural language sentences begins with a deep semantic structure and progresses by applying an ordered series of transformations until a surface string of phonemes or graphemes has been derived. Lexical interpretations are allowed to occur at any stage of the process.

Several linguists (McCawley 1968, Bach 1968, Lakoff 1969) have suggested that the predicate calculus offers a suitable notation for

*This research is supported by the National Science Foundation in its project, Natural Language Research for CAI, Grant GJ 509X.

representing semantic deep structures of natural language statements. The experience of computational linguists in representing textual meanings for computer language processing can be interpreted in support or in denial of this suggestion. On the one hand, several question answering systems have represented sentence meanings as predicate calculus forms. (See Green & Raphael 1969, Coles 1969, Kellogg 1968.) On the other hand most researchers in this field have used attribute-value or semantic network representations. (See Quillian 1970, Carbonnel 1970, Colby et.al. 1969, and Simmons 1968, 70a.) Both approaches have been moderately successful with regard to the goal of answering English questions by computer, and simple English sentences can be transformed with the aid of appropriate lexicons and grammars into either form. Representing very complicated sentences is a process that is equally poorly defined for either form.

In my mind the semantic network representation offers simplicity of graphic and computational representation and easy readability as clearcut advantages over customary predicate calculus notations. In order to preserve meaning it must be quite as precise as the predicate calculus representation of quantification, specification, and the scope of variables. Because of these and other arguments presented previously (Simmons 1970c) I have chosen to represent semantic structures of sentence meaning in the form of networks of wordsense nodes connected by case relations.

The nature and form of transformations used by linguists for generating sentences has been discussed at length in transformational literature. Jacobs and Rosenbaum (1968) present a detailed treatment of the purpose and form of transformations. Friedman (1969) describes a transformational

grammar tester which not only defines the computational form of the transformations but provides generation algorithms for producing phrases and sentences starting from a phrase structure base and applying transformations to achieve the surface structure.

Although various computational linguists have generated coherent sentences and questions from semantic networks (namely, Quillian 1970, Carbonnel 1970, Simmons 1968 and others) their methods have been largely ad hoc and of limited generality.

One recent paper (Woods 1970) has argued for a significant generalization of the linguist's transformational apparatus. Woods represents his English grammar in the form of a state transition network that is augmented with sub-network subroutines and a series of conditions and operations associated with each possible path. He clearly demonstrates that the resulting augmented state transition network is a suitable device for analyzing or generating natural language structures. It is computationally more efficient than the customary form of transformational rules and more powerful; yet it allows the linguist to restrict the power of rules in any way that his theory may require. Woods proves that without restrictions, an augmented state transition network is equivalent in power to a Turing machine.

These ideas of semantic network representations of English meanings and the augmented state transition network representation of transformations are the basis of this paper. In it is presented an algorithm and fragments of grammar for generating sequences of English sentences. Some attention is devoted to methods of assigning pronouns, embedding sentences, and determining voice, mood, aspect and tense of verbs. I believe the approach may serve to suggest to linguistic theorists a useful method for expressing their theories of generative semantics.

## II The Generation Grammar & Lexicon

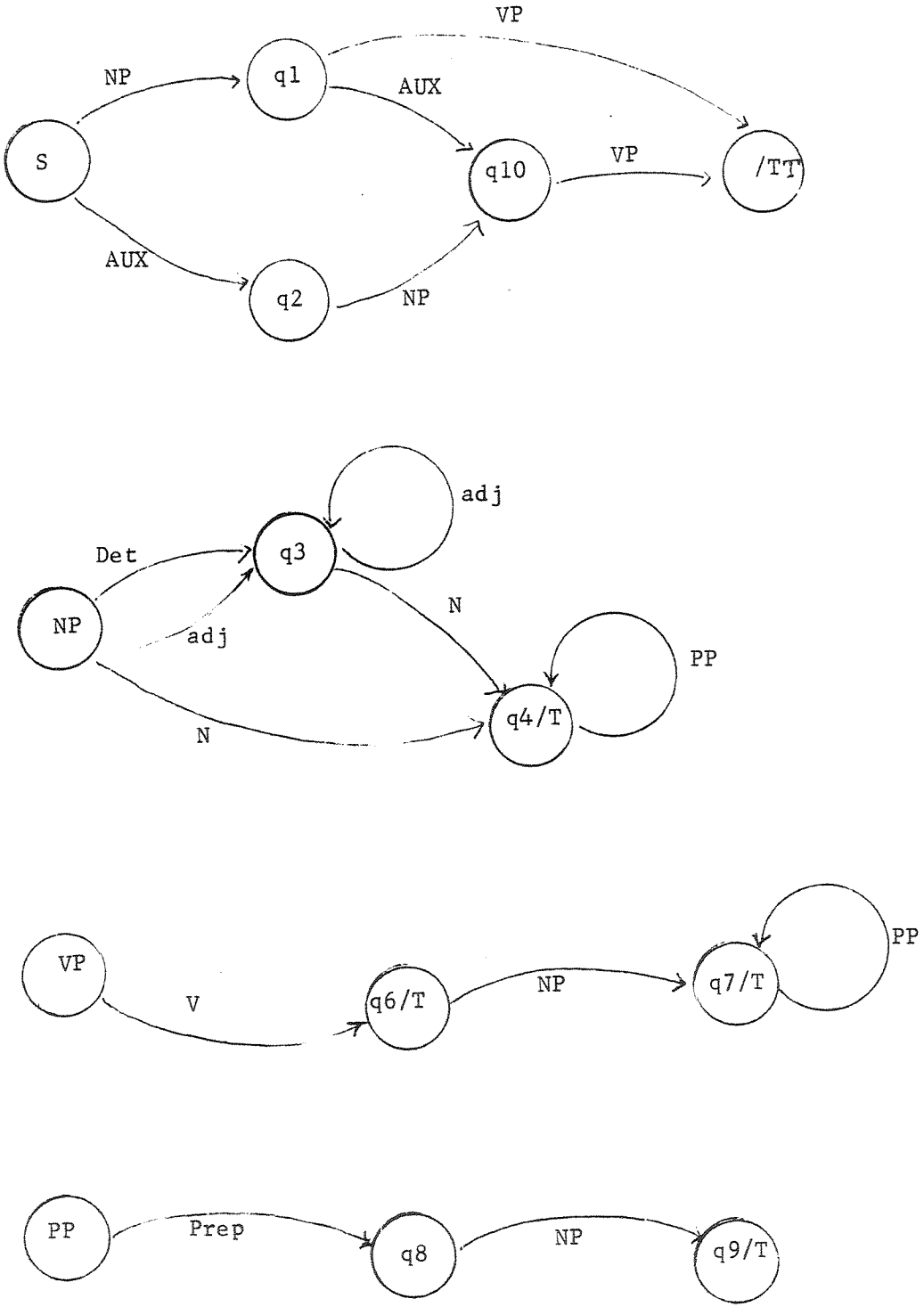
The description of this system of generative semantics will be developed in two sections; this section outlines the form of the grammar and lexicon and describes a random generation algorithm that produces syntactically well-formed nonsense. Section III shows how the semantic nets are used to control the ordering of grammar rules and lexical selections to generate meaningful discourse.

The Grammar: A simple phrase structure grammar can be represented as a state transition network. This can be demonstrated most easily and clearly by an example almost identical to the one Woods (1970, p. 592) used in his illustration of a recursive transition state network. First, the grammar:

$$NP \longrightarrow (DET) + (ADJ^*) + N + (PP^*)$$
$$PP \longrightarrow PREP + NP$$
$$S \longrightarrow NP + (AUX) + VP$$
$$S \longrightarrow AUX + NP + VP$$
$$VP \longrightarrow V + (NP) + (PP^*)$$

This is a context-free phrase structure grammar using the conventions that, parentheses indicate optionality, and the asterisk indicates one or more occurrences. Figure 1 shows the recursive state network representing the grammar. In this graph, the nodes or states are shown as circles with labels such as "S", "NP", "q7", etc., and the arcs or paths are labelled by phrase names or part of speech designators such as aux, prep, n etc. Some states, such as "q7/T", are marked "/T" to indicate possible terminators of the net or subnet in which they occur.

A simple algorithm can be used to generate sentences from this net. We assume that the algorithm can distinguish wordclass names such as




Note: final states marked by "/T"

Fig. 1 Recursive Transition Network
After (Woods 1970)

"det", "adj", "n", etc. from phrase names such as "NP", "VP", "PP", etc. Starting with the symbol "S" we may choose either the "aux" or the "NP" path. Arbitrarily, select "NP". Since it is a phrase name, save the desired state q1 on which "NP" terminates, and find the subnet labelled "NP". This net gives the two choices "det" or "n"; select one, "det" which leads to q3. Since "det" is a terminal wordclass, apply a lexical interpretation to select a determiner from the dictionary; arbitrarily "the". Put "the" as the first element of the output string. Having accomplished the interpretation of that path, the system is now at q3. From q3 there is a choice of adjective or noun paths. Choose adjective, another terminal wordclass; lexically interpret it as "red" to give the output string "the red" and so achieve state q3 again. This time select the noun path labelled "n", interpret it as "wagon" and so achieve state q4. This state is marked "T" to indicate that it is permitted to end an "NP" at this point. Let us do so and discover that the accomplishment of the NP path (as a subnet) led us to state q1, which had been saved previously. From q1, we have only the path "VP" leading to state TT. By following the subnet for VP we might eventuate in a sentence such as "the red wagon broke an axle" and so achieve the S network's terminal state TT, by having generated an NP and a VP.

The reader is invited to explore the net further, starting at S and taking other branches to generate a variety of sentence and question structures.

Table 1 shows a basic LISP algorithm for random generation of sentences from a recursive transition state network grammar. Some of the nonsense sentences generated by this algorithm and the example grammar

GEN ( S )

1. If S is NIL, Return NIL,

2. If S is not an atom
   Concatenate GEN (CAR(S)) & GEN (CDR(S))

3. If S has terminal marker, /T,
   Return NIL if RANDOM NBR is greater than .50
   Otherwise, go on to 4.

4. If S is a word-class name,
   return a randomly selected word of that category

5. Otherwise, GEN(SELECT (PATHS (S)))

Where: CAR returns 1st element of a list

CDR returns the remainder of a list

RANDOM NBR returns a number between 0 and 1

PATHS returns the outgoing path-node pairs from a state

SELECT returns an element randomly selected from a list

Table 1 A LISP-type Algorithm for Random Generation of Sentences from a Network Grammar

are shown in Table 2. In the LISP formalism, the network is represented (on the property list) as a set of 1st,-edge,-3rd triples as follows:

| 1st | Edge | 3rd |
|---|---|---|
| S | NP | q1 |
| S | AUX | q2 |
| q1 | VP | TT |
| q2 | NP | q10 |
| : | | |
| et cetera | | |

The concatenation of recursive calls to the function GEN automatically follows the paths in the network of the grammar, using LISP machinery for keeping track of what states are on the pushdown stack. The lexicon for these examples is assumed to be simply lists of words that are values of the terminal wordclasses "noun", "det", etc.

Now, having seen how the state network is suitable for representing a phrase structure grammar, let us increase its power (still following Woods' development), by associating with each path a set of conditions and operations. The conditions are tests such as those of syntactic or semantic agreement, the presence or absence of lexical features characterizing a form, etc. The operations are transformations of any desired level of simplicity or complexity. If we consider these conditions and operations as functions or subroutines, each of which specifies its set of arguments, we must then indicate what the arguments refer to. For Woods, the arguments are an arbitrary set of registers that contain, as generated, the TYPE of sentence (S or Q), the value of the AUX, of the VERB, of the SUBJECT, of the VP, etc.

In the semantically controlled generator to be described here, these arguments are the relations and their values that characterize a

WILL THE LITTLE RED MAN BREAK A WAGON.

COW WILL PULL WAGON.

WILL MARKET TURN ROAD.

WILL THE ROAD PULL WAGON ON MARKET TO THE RED WAGON.

A MARKET BREAK THE RED MAN.

THE RICKETY RED OLD RED MARKET DID BREAK A RICKETY COW ON THE RED MARKET.

THE OLD MARKET TO A RICKETY COW DID PULL COW.

DID MAN BREAK THE WAGON.

MARKET TO MARKET TURN A RICKETY RED COW.

WAGON TO THE RED ROAD PULL THE MAN.

Table 2 Nonsense Sentences Generated
by Algorithm of Table 1

node in the semantic network. The conditions and transformations associated with ~~a node~~ an arc in the state network are applied to semantic nodes, dividing them, transforming them and finally creating sentence strings from them.

## III Generating English from Semantic Nets

Semantic Nets: The elements or objects of a semantic network for representing discourse meanings are concepts and relations. The primitive concept is a wordsense meaning that is itself represented in the lexicon as an object in a set of relations with other objects. Generally, a concept is a set of relation-concept pairs associated with a labelled object such as C1 in the following illustration:

```
C1 - R1 - C2
     R2 - C3
          .
          .
     Rn - Cm
```

A network is defined as follows:

| | | |
|---|---|---|
| Network | -- | Node* |
| Node | -- | Node + relation set \| L-node |
| relationset | -- | (relation + node)* |
| relation | -- | transformational function |
| L-node | -- | wordsense \| terminal element |

The asterisk signifies one or more repetitions. Nodes are concepts which are sets of relation-node pairs. L-nodes are wordsense references or other terminal elements. The relations are such intersentential connecting meanings as signified by "since", "thus", "because", etc. or deep case relations such as AGENT, OBJECT, DATIVE, etc. which are expressible in the surface string by ordering conventions or particular prepositions. The relations in the net are associated with transformation functions that can form them into syntactic constituents. (For other purposes such as paraphrase, other transformations are also associated with the relations.)

Both discourse and lexicon are represented in semantic net form. The lexicon uses such relations as PRINTIMAGE, SYNTACTIC WORDCLASS, HYPONYM, ANTONYM, IMPLY, etc. Details of the structure for representing

discourse and lexicon are presented in another paper, (Simmons 1970b) and only the minimal description necessary for the understanding of examples is given here.

The semantic representation of a sentence and a portion of the relevant lexical structure are presented in Table 3 to illustrate the relational form of semantic nets. From this table, it can be seen that the sentence "John saw Mary wrestling with a bottle at the liquor bar" has been analyzed into a set of concepts, namely C1-C9, each of which is related by TOK to a lexical wordsense address, L1-L9, where other information concerning the wordsense is located. The concepts are clearly not words--they are named sets of relations with other concepts --but they map into wordsenses that have print images which are words. (In fact, our dictionary specification requires wordsenses to have the relation -WDSNS which maps onto a word which in turn does have print images. This complexity has been omitted from the present example for the sake of a clearer exposition.)

A wordsense may be representable by several words, but it is an unambiguous object of meaning. A concept relates through various relations to a set of other concepts, but it, too, is an unambiguous object. Thus the example sentence is represented as a particular ordered set of unambiguous concepts with explicit relations to each other. This is its semantic representation.

Words such as "saw", "bottle", "bar", etc. obviously each have several sense meanings. The semantic analysis method is assumed to have decided on precisely which wordsense concept is signalled by the choice of a word in context and in this manner mapped a string of words into the network structure of concepts and relations. How this is to be

| | | |
|---|---|---|
| C1 | TOK | L1(see) |
| | TIM | PAST |
| | DAT | C2 |
| | OBJ | C3 |
| C2 | TOK | L2(John) |
| | NBR | SING |
| | DAT* | C1 |
| C3 | TOK | L3(wrestle) |
| | TIM | PROG PAST |
| | AGT | C4 |
| | OBJ | C5 |
| | LOC | C6 |
| | OBJ* | C1 |
| C4 | TOK | L4(Mary) |
| | NBR | SING |
| | AGT* | C3 |
| C5 | TOK | L5(with) |
| | POBJ | C7 |
| | OBJ* | C3 |
| C6 | TOK | L6(at) |
| | POBJ | C8 |
| | LOC* | C3 |
| C7 | TOK | L7(bottle) |
| | DET | INDEF |
| | NBR | SING |
| | POBJ* | C5 |
| C8 | TOK | L8(bar) |
| | NBR | SING |
| | DET | DEF |
| | ASSOC | C9 |
| | POBJ* | C6 |
| C9 | TOK | L9(liquor) |
| | NBR | SING |
| | ASSOC* | C8 |

| | | |
|---|---|---|
| L1 | PI | SEE |
| | PAST | SAW |
| | 3RDP | -s |
| L2 | PI | John |
| | PLUR | -s |
| L3 | PI | wrestle |
| | PAST | -ed |
| | PROG | -ing |
| | 3RDP | -s |
| L4 | PI | Mary |
| | PLUR | -s |
| L5 | PI | with |
| L6 | PI | at |
| L7 | PI | bottle |
| | PLUR | -s |
| L8 | PI | bar |
| | PLUR | -s |
| L9 | PI | liquor |
| | PLUR | -s |

NOTE: Word class relations such as noun, verb, preposition, etc. are parts of the lexicon not illustrated here.

Table 3 Attribute-Value Representation of Example Discourse and Lexicon

accomplished is the subject of another paper (Simmons 1970d and of discussions by Katz (1965), Leech (1970) and Quillian (1970)). Here we will accept it as given by a human analyzing the sentence with his intuitive understanding.

This semantic analysis is partially represented by the graph of Figure 2. This figure suggests how words in a sentence activate cascades of meaning in the lexical nets. Not shown, are the definitional and implicational relationships that characterize each lexical entry for use in paraphrase generation. If these were considered, the complete meaning of a sentence would be represented as that portion of the concept net that it activates--presumably a very large subnet, indeed.

Our purpose here, however, is not primarily to show how the semantic network represents meanings but instead to demonstrate how it can be used in conjunction with a grammar to generate sequences of coherent English sentences. Let us first look at a simple sequence generator that will transform a concept into a sequence of words representing its set of relational pairs. The relational pair is referred to as a relation and its argument node. The sequence generator is an algorithm that translates relations and nodes into their lexical representation. For this example, it will represent relations in the strings it generates in exactly the form they are shown in Table 3; i.e., ASSOC as ASSOC, AGT as AGT, POBJ as POBJ, etc. If the argument node of a relation is an L-node, it prints the PI or print image value; otherwise it expands a C-node, recursively. This sequence generator will ignore such relation pairs as TIM - PAST, DET - Def/indef, NBR- etc.

If we begin with this algorithm at C1, it will generate the following:

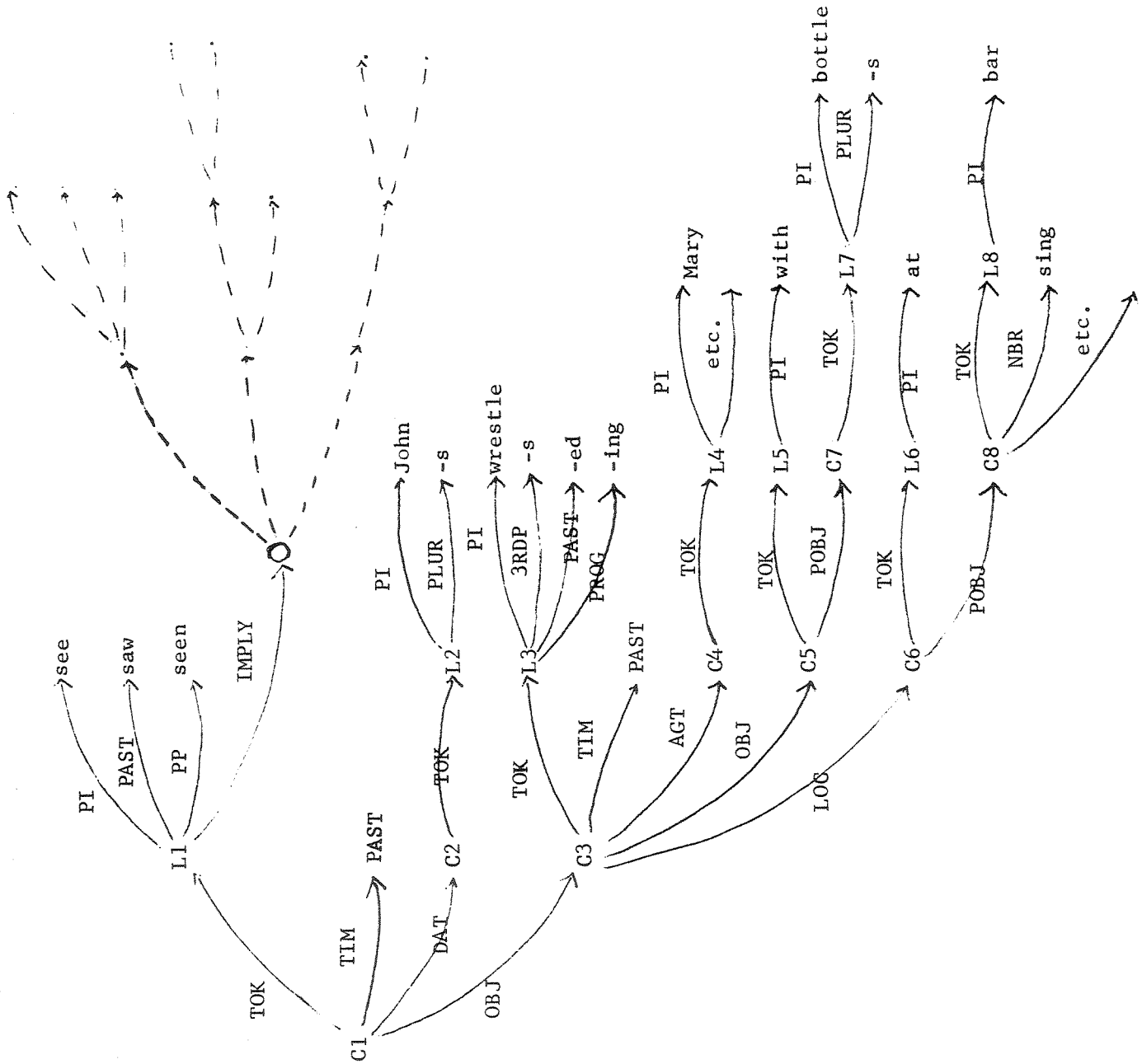



Fig. 2 Partial Graph Representation of Table 3

SEE DAT JOHN OBJ WRESTLE AGT MARY OBJ WITH POBJ BOTTLE
LOC AT POBJ BAR ASSOC LIQUOR

This string of words and symbols carries much of the meaning of the original sentence and the simplicity of its generation immediately explains one aspect of the popularity of semantic nets with computational linguists. That is, the syntax of the semantic nets is not vastly dissimilar to that of English phrase constructions. If some of the word-sense nodes map onto more than one English word, additional strings such as "OBSERVE DAT JOHN..." and "...OBJ STRUGGLE AGT MARY..." will also be generated. This shows one level of paraphrase capability inherent in semantic structure--alternate lexical representations of wordsense concepts.

Semantic Generation: We have previously seen that if we apply a syntactic generator to a grammar and lexicon, we obtain nonsense sentences that are syntactically well-formed strings of English words. Here, in simply generating a printable linear representation of a semantic net, the result is a meaningful string of words that is not syntactically well-formed. By applying a set of syntactic transformations (i.e. a grammar) to the semantic relation pairs, the generator can impose the syntactic structure of English onto the lexical interpretation process and so produce well-formed sentences of English.

The key feature of this process seems to be that a recursive transition state network (or for that matter any grammar with transformations) is an ordering of a series of transformation rules. The semantic network is also a complex ordering of semantic relations that conjoin concepts. The semantic relations of discourse are deep case relations and

other relational meanings that map onto syntactic relations and relational words in the surface string. Thus, AGT, DAT, INST, SOURCE can map onto the syntactic relations of SUBJECT, OBJECT, and various prepositional phrase COMPLEMENTS in accordance with an ordering imposed by the grammar network. Various syntactic rules--i.e. paths in the network--offer differing choices of surface strings to represent the relational ordering of concepts in the semantic network. The transformation from semantic to syntactic relation is accomplished by associating as part of the content of each semantic relation, the transformation that forms it and its concepts into a syntactic constituent of a sentence.

The Grammar: The form of the grammar rule is a node-arc-node triple where the node is a label and an arc is the name of a function which may perform various tests and transformations as needed. The arc is also necessarily the name of a relation in the semantic net and, for uses in question answering and paraphrasing,it may be associated with additional functions for accomplishing inferences. A formal definition is as follows:

Grammar Rule := Node + Arc + Node

Node := Label

Arc := Function Name ≡ Semantic Relation Name

The labels are any arbitrary sequence of symbols but the arcs of the grammar must correspond to the names of semantic relations in the semantic network. The grammar is tabularly represented as an ordered set of triples as in Figure 3, where the symbol -/- signifies an unlabelled or unconditional arc and T a terminal state label. The relation of the triples to a graph can be seen in Figure 4a where a partial state graph of the

grammar is shown. Figure 4b shows an equivalent set of rewrite rules using the conventions that parentheses indicate optional elements and braces a choice of one or more elements. All symbols in Figure 4b that are in the right half of the rules are transformations that are defined by LISP functions. (This use of actual LISP functions is in contrast to Woods' (1970) presentation of a special language for accomplishing transformations).

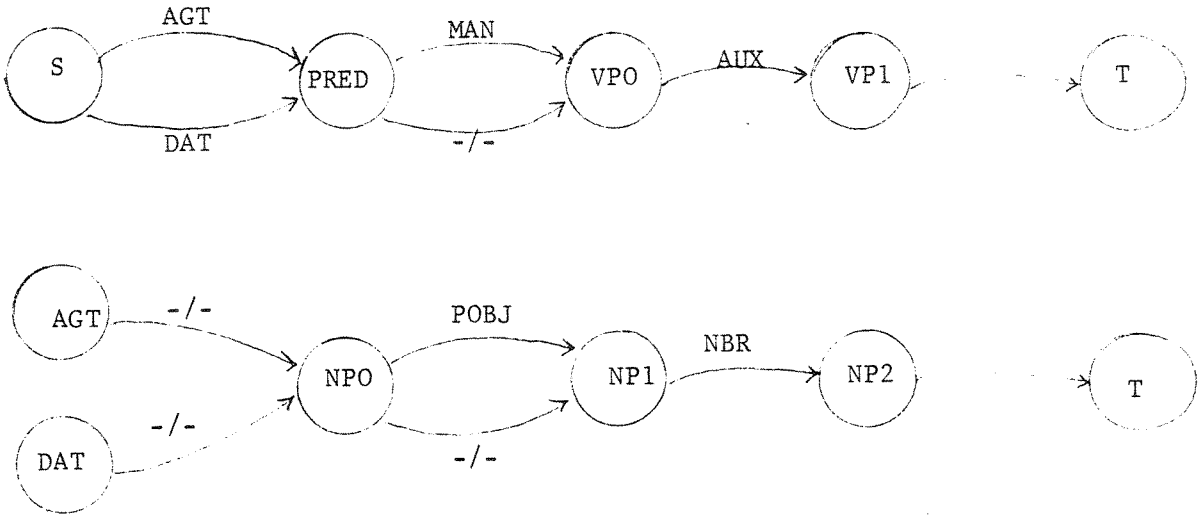
At each node in the transition net grammar, a choice of paths is typically available. In the random generation example the function SELECT was used to make a random choice. In generating meaningful sentences from a semantic net, the semantic relations perform this choice function. That grammar path is chosen whose name corresponds to a semantic relation in the concept being generated. The procedure will become clearer in the labor of generating a sentence.

An Example: For a simple example, the semantic structure of Table 3 will be used in conjunction with the grammar illustrated in Figure 3 to generate a complex sentence. The procedure begins with the selection of a concept from the net as a starting point. This concept is labelled S; i.e. the relation pair LAB-S is added to the concept. The example starts with concept C1 from Table 3.

| | | |
|---|---|---|
| S | AGT | PRED |
| S | DAT | PRED |
| AGT | -/- | NPO |
| DAT | -/- | NPO |
| NPO | POBJ | NP1 |
| NPO | -/- | NP1 |
| NP1 | NBR | NP2 |
| NP2 | DET | NP3 |
| NP2 | -/- | NP3 |
| NP3 | MOD | NP3 |
| NP3 | -/- | NP4 |
| NP4 | NS | T |
| PRED | MAN | VPO |
| PRED | -/- | VPO |
| VPO | AUX | VP1 |
| VP1 | VS | VP2 |
| VP2 | POBJ | VP3 |
| VP2 | -/- | VP3 |
| VP3 | LOC | VP4 |
| VP3 | DAT | VP4 |
| VP3 | INST | VP4 |
| VP3 | -/- | VP4 |
| VP4 | -/- | T |
| OBJ | POBJ | NP1 |
| OBJ | NBR | NP2 |
| OBJ | AGT | PRED |

Note: The relations in the middle columns above are directed arcs between the nodes (states) in the left-side columns and the nodes in the right-side columns; the arcs denote transformations used in the generation algorithm.
Unlabelled arcs (here designated by -/-) denote unconditional transitions to the right-side state and T is a terminal state.

Figure 3. Grammar for Generating a Sentence from Semantic Net of Figure 1




4a. A Fragment of State-Graph Corresponding to the Grammar of Figure 3

---

$$S \longrightarrow \left\{\begin{matrix} AGT \\ DAT \end{matrix}\right\} + PRED$$

$$AGT \longrightarrow NPO$$

$$DAT \longrightarrow NPO$$

$$NPO \longrightarrow (POBJ) + NBR + (DET) + (MOD) + NS$$

$$PRED \longrightarrow (MAN) + AUX + VS + (OBJ) + \left(\left\{\begin{matrix} LOC \\ DAT \\ INST \end{matrix}\right\}\right)$$

4b. Rewrite Rules Corresponding to Grammar of Figure 3

---

Note: Symbols are explained in text.

Figure 4. Alternate Representations of Grammar of Figure 3

C1 is labelled S and the grammar of Figure 3 is entered to find the relevant node (or rule) beginning with that symbol. The first relevant path, S-AGT-PRED is attempted. The structure, C1, is examined to see if the relation AGT is present. It is not, so the next path, DAT-PRED, is checked. The relational pair, DAT-C2 is found on structure C1 showing that this path through the grammar is possible. The value of that pair, structure C2, is now labelled DAT. C1 is relabelled PRED (for future reference) and the grammar is entered at the DAT node. The relevant path is DAT-/-NPO. The symbol "/" indicates an unconditional relabelling corresponding to a unary rewrite rule. Structure C2 is thus relabelled NPO and the grammar is entered at the NPO node. NPO offers the path POBJ-NP1 and /-NP1 showing that the POBJ path is optional --if present in the structure it indicates a prepositional phrase and will be followed; if not present the structure will be unconditionally labelled NP1. The relation POBJ is not present in C2, so C2 is labelled NP1, and the grammar is entered at that point. The grammar node NP1 offers only the path, NP1-NBR-NP2. It can only be followed if the relation NBR is found on the structure as in fact it is on C2 in the pair, NBR-Sing.

NBR is a relation that has associated with it a series of tests and transformational operations that can be stated briefly as follows:

1) if value of NBR is Sing
   set NS (Noun String) to the value of the Print Image of TOK
2) if value of NBR is Plural
   set NS to the MORPH value for TOK and Plural
3) Add the relational pair NS and its value to the concept.

MORPH is a function of the two arguments which are the lexical values of TOK of the coneept and of that lexical entry's plural form. When

operated with these arguments it returns the print image of the plural form of the word. Since the system being described is realized in LISP, NBR is defined as a LISP function to accomplish the tests and operations. The result of operating NBR as a function was to add to C2 the relation NS with the value "JOHN"; then C2 is relabelled NP2, ~~and to relabel it as NP2.~~

Looking up NP2 in the grammar, we see that it has an optional DET path to NP3, which in its turn has an optional MOD path to NP4. Since neither a DETerminer or any MODifiers are present in the semantic structure, the traversing of these paths results in putting the concept at state NP4--i.e. its relabelling as NP4. NP4 (for this simple grammar) leads by the relation NS (noun string) to a terminal state, T. The relation NS is also a function that has the effect of concatenating its value, "JOHN" to the output string being generated. Since the terminal state, T, was reached in the generation of C2, we have now managed to cross the DAT path and achieve the state PRED which had been attached as a label to C1 after using its DAT relation.

PRED offers the optional path, MAN(ner adverb) -VPO which is not realized in C1, so C1 is relabelled VPO via the unary rule "-/-VPO". This node gives only an AUX-VP1 path. The pair AUX-Past is found on C1, and the function AUX prepares a verb string relation VS whose value for this example is "SAW". and attaches the pair VS-SAW to the structure C1. Now C1 is relabelled VP1. ~~This relabels C1 as VP1~~. The path VP1-VS-VP2 then places the verb string on the output string and takes C1 to state VP2.

Node VP2 calls for an OBJ relation which is found on C1 in the pair OBJ-C3. C1 is relabelled VP3 for future use after C3 has been developed, and C3 is labelled OBJ. Three paths are offered at node OBJ, namely, POBJ-NP1, NBR-NP2, and AGT-Pred. The first two are for the

cases where the object is a prepositional phrase or a noun phrase; the third is for an embedded sentence. C3 is found to have the relation pair, AGT-C4, so it is labelled PRED and C4 is labelled AGT, and the procedure continues recursively, adding the string "Mary wrestling with a bottle at the liquor bar" to the already generated fragment "John saw". When C3 has been generated, control returns to C1 which had been labelled VP3. Since C1 has no further relation pairs, the state T is quickly achieved via VP4 and the process ends with the completed sentence:

JOHN SAW MARY WRESTLING WITH A BOTTLE AT THE LIQUOR BAR.

Additions to the grammar of Fig. 3 can be devised to develop syntactic variations of the example sentence and to generate the sentences that result from starting at Concept C3 or C7 and exhausting the concept network. Whatever special conditions and transformations are required can be associated with the names of arcs in the grammar.

The Generation Algorithm: The algorithm and data structures that accomplish semantically controlled generation are interesting computational structures particularly in view of their simplicity of expression in LISP. Table 4 shows a basic LISP algorithm, called GEN. It has been set up as a sequential program (rather than a more elegant expression) to aid its readability. Notes to this table explain the functions called in the algorithm.

The grammar network is placed on the LISP property list in the following form:

(S (GR ((AGT PRED) (DAT PRED) ( etc.) (etc. ) ) ) )

```
      (GEN (LAMBDA (ST GR) (PROG ()
1.    (COND ((CSETQ J (GET ST TERM)) (RETURN (LIST J)))
2.    ((NULL ( CSETQ (GET ST LAB))) (RETURN NIL)))
3.    (CSETQ J (GET J GR))
4.    (COND ((NULL J) (RETURN NIL))
5.    ((UNARY (1STRUL J)) (PROG2 (PUT ST LAB (2ND J))
                                 (RETURN (GEN ST GR))))
6.    ((NULL (CSETQ K (GET ST (1ST J))))
                        (PROG2 (CSETQ J (CDR J)) (GO 4))))
7.    (PUT ST LAB (2ND J))
8.    (COND ((OR (TERM K) (INVRS (1ST J)))
                      (RETURN (GEN (APPLY (1ST J) (LIST K)) GR))))
9.    (PUT K LAB (1ST J))
10.   (DELETE ST (1ST J))  (APPLY (1ST J) (LIST K))
11.   (RETURN (APPEND (GEN K GR) (GEN ST GR)))  )))
```

---

Notes: GET of a STructure name and a Relation returns the value of the the relation.

CSETQ (J K) sets J to the value of K, and returns K as its value.

NULL is True for the value NIL, False otherwise.

1STRUL returns the first of a set of rules.

1ST returns the arc of the first rule: 1ST(S NP VP)→ NP
2ND returns the state value: 2ND(S NP VP)→ VP

UNARY tests for a unary rule as (AGT -/- NP)

PUT puts a relation value pair on a structure.

APPLY operates a relation as a LISP function with a list of arguments. (In Texas LISP APPLY does not require an a list.)

DELETE deletes the relation from its structure and the backlink from the structure that it refers to.

TERM tests if its argument is a terminal print image.

INVRS tests for an inverse arc as AGT*, OBJ*, etc.

---

TABLE 4. Algorithm for Semantically Controlled Generator

The function, (GET S GR) returns the list of path-node pairs leading away from the state S. The semantic structure network, also on the LISP property list, is in a similar form, as follows:

(C1 (DAT C2) (TIM PAST) (TOK L1) (OBJ C3) )

The GET of a structure and a relation, i.e. GET(C1 DAT), returns the value associated with the relation, i.e. C2. The relational terms such as AGT, DAT, OBJ, TIM, etc. are each defined as small LISP functions that test for appropriate conditions and apply transformations to the structure. As LISP users know, the use of the property list feature in this system maximizes the efficiency of storage and retrieval operations in the language.

The algorithm is conveniently described in eleven steps corresponding to the numbered sets of statements in Table 4. ST refers to the starting semantic structure node; GR is the grammar to be consulted.

1. Tests to determine if a TERMinal print image has been developed as the value of TERM, if so GEN returns its value as a list.
2. Tests to determine if the semantic STructure has been labelled, if not, GEN returns NIL; if so, J is the label.
3. The value of J is taken as an entry point to the grammar and J is reset to the grammar rules if any.
4. If J is NIL, no grammar path or none corresponding to a semantic relation (see steps 5 and 6) is available and GEN returns NIL signifying a grammar that is inadequate for generating sentences from the structure. (Instead of NIL, a function could be used here to go into a backup procedure for seeking some alternate path through the grammar; as yet we do not do this.)
5. If the rule obtained is unary (as AGT -/- NP), the structure is relabelled and GEN is called recursively with the relabelled

structure. (Equivalent to looping back to step 2 with the relabelled structure).

6. In (GET ST (1ST J)), 1ST J is the syntactic arc and this GET call seeks to determine if this arc is also a semantic relation in the structure ST. If not--that is GET returns NIL--the remainder of the set of grammar paths--i.e. (CDR J)-- is examined by looping back to Step 4.

7. When the first 6 steps have been accomplished successfully--i.e. there is a grammar rule corresponding to a semantic relation in the structure--the structure ST is relabelled by the 2nd state--(2ND J); with the rule,S AGT PRED, ST would be relabelled PRED.

8. This is the step that eventually operates the functions associated with the arcs. In Step 6, K was set to (GET ST (1ST J)). 1ST J is the semantic relation and K becomes its value. If the value of K is terminal-- as in the relational pairs, TOK-house, NBR-PL, VOICE-ACTIVE, etc. as contrasted with non-terminal as in AGT-C2, OBJ-C3, MOD-C8, etc.--, or if the semantic relation is an inverse as in AGT*-C1, OBJ*-C8, etc*, THEN the arc is applied as a function with the ST structure as an argument-- (APPLY (1ST J) (LIST ST))- and GEN is called recursively on the resulting transformed structure.

9. If the condition of Step 8 is not true, structure K is labelled by by the arc name

10. The relation pair giving rise to K is deleted from ST, and the function associated with the arc is applied.

---

* See Section IV for the use of inverse relations.

11. The value of GEN is the GEN of the new structure K, concatenated with the GEN of the modified original ST structure. For example if the rule NP5 MOD NP6 had been applied successfully to C5 TOK house

MOD C6

OBJ* C14

with the call, (GEN C5 GR1), the result would be (Append (GEN C6 GR1) (GEN C5 GR1)) where MOD-C6 was deleted from C5. As each path in the semantic net reaches a terminal semantic state (described in Step 8), the functions associated with the arc produce print images as the value of TERM on the structure.

This is a highly recursive function that terminates when every call to GEN has returned either terminal print images or nils. The concatenation of these results is the output sentence that GEN generates. It will be seen in Section IV that because of the presence of backlinks (inverse relations on a semantic node such as common references to nouns from different parts of the network), a call to GEN with a single concept structure as its ST argument will usually generate a whole discourse-set of sentences rather than a single sentence. How long or short the sentences are to be is controlled by the grammar and by some decision functions that are mentioned in the next section. What is important to notice here, is that this one algorithm is the primary mechanism for allowing a semantic net to limit and control what sentence structures a grammar can generate.

Generating Form Determination: The correct tensing of natural language sentences is an important requirement on a semantically controlled generation process. In the present system this too is controlled by the interaction of semantic structures and the grammar. Form determiners in English include Voice, Form, Tense, Aspect and Mood. For semantic net structures these are taken as relations whose values have been set by some largely stylistic plan that organized a subnetwork of concepts to be generated.* Such a plan might take a semantic net such as

| | | |
|---|---|---|
| C1 | TOK | build |
| | AGT | (John)** |
| | OBJ | (house) |

and augment it with form determiners as follows:

ST 1)

| | | |
|---|---|---|
| C1 | TOK | build |
| | VOICE | passive |
| | FORM | progressive |
| | ASPECT | perfect |
| | TENSE | future |
| | MOOD | interrogative |
| | AGT | (John) |
| | OBJ | (house) |
| | LAB | S |

This structure carries the information that will lead to the generation of the question, "Will the house have been being built by John?" under the control of the following grammar:

| | | | | | |
|---|---|---|---|---|---|
| S | VOICE | TFM | | | |
| TFM | FORM | ASP | INDIC | -/- | S1 |
| ASP | ASPECT | TNS | INTERROG | PREVB | S1 |
| TNS | TENSE | PROP | S1 | SUBJ | PRED |
| PROP | MOOD | T | SUBJ | -/- | NPO |

* We are very much interested in designing such a device, but our first attempts at it showed the necessity of developing a very reliable semantically controlled generator as a first step to show what kind of preliminary transformations are required to organize a set of semantic nets for straight-forward generation, and the stylistic plan as a research area still remains to be studied.

** Parentheses are used to abbreviate the semantic structure. AGT-(John) is actually AGT-C2, C2 TOK John, etc.

The grammar is continued with the rules of Figure 3 which have been presented earlier.

The generator is called with the structure C1 and the grammar shown above. The first rule, S VOICE TFM is obtained by the generator from the structure's Label, LAB-S. This rule applies the function named VOICE to C1 and the structure is then relabelled TFM.* The result of applying VOICE to C1 which has the relation pari VOICE-passive is to determine SUBJect and OBJect values, to set up a verbstring VS, and to delete the relation VOICE-passive, as follows:

ST 2)

| | | |
|---|---|---|
| C1 | VS | (BE/built) |
| | FORM | progressive |
| | ASPECT | perfect |
| | TENSE | future |
| | MOOD | interrogative |
| | SUBJ | (house) |
| | OBJ | (by John) |
| | LAB | TFM |

The rule associated with the value of the label, TFM is then obtained from the grammar and applied. This rule reads, TFM FORM ASP. The relation pair FORM-progressive is found in the structure, so the transformation function associated with FORM is operated. The effect of the function FORM is to further modify the verbstring, VS, and to delete the pair FORM-progressive. The structure is then relabelled to give LAB-ASP and appears as follows:

ST 3)

| | | |
|---|---|---|
| C1 | VS | (Be being built) |
| | ASPECT | perfect |
| | TENSE | future |
| | MOOD | interrogative |
| | SUBJ | (house) |
| | OBJ | (by John) |
| | LAB | ASP |

In a similar fashion the next three rules,

| | | |
|---|---|---|
| ASP | ASPECT | TNS |
| TNS | TENSE | PROP |
| PROP | MOOD | T |

* Note that these are unary transformations as discussed earlier. See step 8 of Table 4.

are then successively applied (see Table 6 for detail) to result in:

| | | | |
|---|---|---|---|
| ST 6) | C1 | VS | (have been being built) |
| | | PREVB | will |
| | | SUBJ | (house) |
| | | OBJ | (by John) |
| | | LAB | INTERROG |

The operation of TENSE in addition to adding auxiliaries, modifies the first verb in VS to agree in number with the subject.

The rule, INTERROG PREVB S1, now applies followed by the rule, S1 SUBJ PRED. The function associated with PREVB begins the output string with "Will" concatenated to the output string generated by S1. The values of SUBJ and OBJ have been shown in parentheses to indicate that they are in fact structures that will have to be generated by NP rules shown in Figure 3. When this has been done by a series of rules starting from S1 the resulting sentence is the desired form:

"Will the house have been being built by John?"

This illustration of the sequence of structures developed in the application of the rules for form determination should serve to emphasize the facts that 1) each arc in the grammar can be a transformational rule, 2) the grammar controls the sequence in which the rules are operated, and 3) the semantic structure selects the path through the grammar (by the relation pairs that comprise its content).

The effect of various values of the form determiner relations (i.e. VOICE active/passive, TENSE present/past/future, etc.) can be seen in the following detailed computational model which represents the essential operations embodied in the transformation functions, VOICE, FORM, ASPECT, etc.

Computational Structure of Voice, Mood, Form & Tense: The model for this syntactic structure is a set of three relational pairs:

SUBJ-Value, VSTRING-Value, and OBJ-Value. The VSTRING value is a list or a pushdown stack which allows addition or deletion of first members. The effect of Voice, Form, Aspect, Tense and Mood in that order is to accumulate the values for these three relational pairs. An example will develop the complex forms of the previous illustration:

Will the house have been being built by John?

Form determination is signalled by the following values:

Voice - Passive
Form - Progressive
Aspect - Perfect
Tense - Future
Mood - Interrogative

VOICE first operates to put the verb which was the value of TOK onto VSTRING and to assign values to SUBJ and OBJ. SUBJ is assigned the value of any relations AGT, DAT, INSTR in that order of preference. OBJ is assigned the value NIL in case there is no relation, OBJ in the semantic structure. If the value of VOICE is passive, then the verb on VSTRING is replaced by its -EN form (i.e. past participle), BE is added to the first position (henceforth called FIRST) on VSTRING, the values of SUBJ and OBJ are exchanged, and the new value of OBJ is adjusted to take the preposition "by".

FORM with the value Progressive, changes the verb on FIRST to the -ING form, making BEING and puts BE on FIRST. (We now have BE BEING BUILT). ASPECT applies Perfect, by changing the verb on FIRST to its -EN form and putting HAVE on FIRST: TENSE-future is then applied to put WILL on FIRST thus forming a verbstring of WILL HAVE BEEN BEING BUILT with a subject, HOUSE and an object, BY JOHN. (TENSE-past would have changed HAVE on FIRST to HAD). TENSE also determines the grammatical number on the structure which is the value of SUBJ and modifies the verb in FIRST to agree. MOOD, with an Interrogative value assigns the

relation PREVB the value of FIRST, and Labels the structure INTERROG so that a rule, INTERROG PREVB S1, begins a path that generates: WILL JOHN HAVE BEEN BUILDING THE HOUSE.

This example is illustrated in Table 6. The effects of the transformations signalled by Voice, Form, etc. cumulate, one on another as seen in the Table 6 example. Certain combinations such as Passive-emphatic and Future-emphatic are not allowed in English. We have also noticed that modals such as "may", "might" etc, appear to follow the same paradigm but have not yet studied their behavior in detail.

| CONDITION | RULE | RESULT |
|---|---|---|
| VOICE-ACTIVE | Put verb on FIRST<br>Set SUBJ & OBJ values | V STRING: FIRST = BUILD; REMAINDER = ; REGISTERS: SUBJ = John, OBJ = house |
| PASSIVE | Add -en to FIRST<br>Put BE on FIRST<br>EXCHANGE SUBJ & OBJ<br>Add BY to OBJ | BUILD + EN \| ; BE \| built; REGISTERS: SUBJ = house, OBJ = by John |
| FORM-PROGRESSIVE | Add -ing to FIRST<br>Put BE on FIRST | BE + ING \| built; BE \| being built |
| ASPECT-PERFECT | Add -en to FIRST<br>Put HAVE on FIRST | BE + EN \| being built; HAVE \| been being built |
| TENSE-FUTURE | Put WILL on FIRST<br>Modify verb on FIRST<br>to agree in NBR with subject | Will \| have been being built |
| MOOD-INTERROGATIVE | FIRST + SUBJ + VSTRING REMAINDER + OBJ | will + the house + have been being built + by John |



Table 6 Example of Sentence Form Determination

# IV MULTI-SENTENCE DISCOURSE

The structure of semantic networks has been developed to represent aspects of the meaning of multiple sentence texts in such a manner as to enable the answering of questions, the generation and evaluation of paraphrases, and for the even more primary purpose of providing a research vehicle for the study of such complexities of language as coherence, intersentential connections, conditions on embeddings, and rules for pronominalization, anaphora, and ellipsis. Experience with these areas of linguistic research has strongly suggested that the discovery of generative procedures is the most profitable approach to understanding how to analyze the examples of their occurrance in text.

The following short multi-sentence discourse was composed as a basis for study.

> "John saw Mary wrestling with a bottle at the liquor bar. He went over to help her with it. He drew the cork and they drank champagne together."

This brief discourse illustrates some common uses of pronominalization, anaphora, ellipsis, embedding, etc. The development of semantic coding conventions, grammar rules, and generator control features for generating various multi-sentence representations of this discourse has already proved highly instructive.

In this section some of the methods and rules for generating multi-sentence discourse will be described briefly and with special emphasis on the questions and problems that arise.

Generating Sequences of Sentences: The discourse example is represented as a semantic net in its attribute-value form in Table 5. This net can be seen to encode much more detail than the example in Table 3, especially with regard to voice, mood, form and tense of verb concepts. The AUX relation is used in the semantic net to record the relative beginning and ending times of verb events as a pair of numbers. This additional information was found to be essential for decisions regarding the embedding of sentences. The backlinks from each concept to concepts that refer to it are shown explicitly as AGT*, INST*, POBJ*, etc. as these are also crucial to the sequence of sentences in a discourse. The lexical references are foreshortened in this example to show the value of TOK simply as the print image associated with the appropriate wordsense.

As far as is represented in the semantic net, there are no sentences; only concept structures. Forming linear or embedded sequences of sentences is a matter controlled by the grammar and the generation procedure. The generator operates on a list of discourse nodes. During the generation process nodes may be added to or deleted from this list. The back references, AGT*, OBJ*, etc, are the semantic relations that key the process. Each time a structure Si refers to another structure Sj by relation R, Sj refers back to Si by R*, the inverse of R. When a noun phrase is generated as an Agent or Object of a verb structure, the inverse relation (denoted by an asterisk concatenated to the relation name) is deleted; otherwise it would generate an infinite sequence such as : JOHN WHO SAW WHO SAW... This kind of generation would follow from a structure such as:

| | | |
|---|---|---|
| E1 | MOOD | INDIC |
| | TENSE | PAST |
| | FORM | SIM |
| | VOICE | ACT |
| | OBJ | E3 |
| | AGT | E2 |
| | AUX | (1 2) |
| | TOK | SEE |
| E2 | TYP | SING3 |
| | AGT* | (E15 E12 E10 E1) |
| | NBR | S |
| | TOK | JOHN |
| E3 | MOOD | INDIC |
| | TENSE | PAST |
| | FORM | PROG |
| | VOICE | ACT |
| | OBJ* | E1 |
| | LOC | E7 |
| | DAT | E5 |
| | AGT | E4 |
| | AUX | (0 3) |
| | TOK | WRESTLE |
| E4 | TYP | SING3 |
| | OBJ* | E12 |
| | AGT* | E3 |
| | NBR | S |
| | TOK | MARY |
| E6 | TYP | SING3 |
| | NBR | S |
| | DET | A |
| | TOK | BOTTLE |
| E8 | TYP | SING3 |
| | MOD | E9 |
| | NBR | S |
| | DET | THE |
| | TOK | BAR |
| E9 | DEG | POS |
| | MOD | E8 |
| | TOK | LIQUOR |
| E10 | MOOD | INDIC |
| | TENSE | PAST |
| | FORM | SIM |
| | VOICE | ACT |
| | LOC | E11 |
| | AGT | E2 |
| | AUX | (2 3) |
| | TOK | GO |
| E11 | INF | E12 |
| | LOC | E10 |
| | TOK | OVER |
| E12 | MOOD | INDIC |
| | TENSE | PAST |
| | FORM | SIM |
| | VOICE | ACT |
| | IOBJ | E5 |
| | OBJ | E4 |
| | AGT | E2 |
| | AUX | (2 3) |
| | TOK | HELP |
| E15 | MOOD | INDIC |
| | TENSE | PAST |
| | FORM | SIM |
| | VOICE | ACT |
| | OBJ | E21 |
| | AGT | E2 |
| | AUX | (3 4) |
| | TOK | DRAW |
| E16 | MOOD | INDIC |
| | TENSE | PAST |
| | FORM | SIM |
| | VOICE | ACT |
| | MAN | E20 |
| | OBJ | E22 |
| | AGT | E18 |
| | AUX | (4 5) |
| | TOK | DRINK |
| E18 | SECN | E4 |
| | FIRN | E2 |
| | AGT | E16 |
| | TOK | AND |
| E20 | MAN | E16 |
| | TOK | TOGETHER |
| E21 | TYP | SING |
| | POSS | E6 |
| | OBJ | E15 |
| | NBR | S |
| | TOK | CORK |
| E22 | TYP | SING |
| | OBJ | E16 |
| | NBR | S |
| | DET | THE |
| | TOK | CHAMPAGNE |

Table 5 Semantic Representation of a Multi-sentence Discourse (prepositions omitted)

| | | |
|---|---|---|
| C1 | TOK | SAW |
| | AGT | C2 |
| C2 | TOK | JOHN |
| | AGT* | C1 |

and a grammar containing rules such as:

| | | |
|---|---|---|
| S | AGT | PRED |
| AGT | TOK | NP4 |
| NP4 | AGT* | NP5 |

where AGT* was a transformation embedding the sentence generated from its value (C1). Deletion of the backlink AGT*-C1 at the time of generating C2 as an Agent, prevents this loop.

The grammar in fact contains rules such as the following to provide for embedding:

| | | |
|---|---|---|
| NP5 | OBJ* | NP6 |
| NP5 | AGT* | NP6 |
| NP5 | INST* | NP6 |
| NP5 | -/- | NP6 |

These allow for the optional embedding of a sentence in which the structure labelled NP5 is an Agent, Object or Instrument. This is accomplished by having the backlink relations name functions which accomplish appropriate embedding transformations.

Each time the option of an embedding is considered, a general embedding function is called. The result of it and the specific embedding functions (AGT*, OBJ* etc.) is to:

1) compare the time of the verb concept to be embedded with the event time of the dominating sentence (i.e. compare the values of AUX; see Table 5)

2) determine the form of embedding as relativization, and its pronoun, or adverbial clause introduced by BEFORE, DURING, or AFTER in accordance with the temporal relation of the two clauses.

3) determine if an embedding is allowed according to certain style limits such as depth and complexity (where depth is a number showing how many embeddings within embeddings are allowed and complexity is the total number of embeddings allowed in a sentence.)

4) finally use a random number, probability .5, decision as to whether to embed or not, if otherwise possible; and, if an adverbial clause, whether it should be put in a pre-position, embedded, or post-positioned.

5) terminate the NP and put the new verb concept on a list of sentences still to be generated, or cause the generation of a sentence or clause to be embedded according to the conditions determined above.

It is worth mentioning that this complexity of decision process requires less space as LISP conditionals then to describe in English. In the event that the decision is to generate an embedding, the choices of relative pronoun or adverbial introducer are recorded as relational pairs on the semantic structure, and the generation algorithm continues its process. If no embedding is to occur, the value of the backlink is placed on the list of structures to be generated as sentences later in the discourse. As each sentence is generated, its starting structure node is deleted from the list.

The significant insight that this study of embedding has given us is that, except for embedding stative verb structures, the temporal relations between two events one to be embedded in the other, is of paramount importance. Consider the following sets of examples:*

*a) John who drew the cork went over to help Mary.

*b) John who went over saw Mary wrestling...

c) John who saw Mary wrestling... went over...

?d) John who drew the cork and Mary whom he helped drank champagne together.

---

*Where the temporal sequence of events (from Table 5) is: see -1, wrestle -2, went -3, help -3, drew -4, drank -5. Note that "John went over to help" signifies that "to help" is an intention of John that apparently coincides in time with his "going over".

and,

a') John before he drew the cork went over to help...

b') Before he went over... John saw Mary...

c') John went over... after he saw her wrestling...

c') ????

The question marks associated with d and d' suggest that it is usually awkward to separately modify the elements of a conjunction. The contrast of the a b c cases with the a' b' c' examples show that unless the temporal sequence of events is preserved in the surface sequence of verbs, a time relation adverb must be used to signal the temporal sequence.

These remarks and examples give some idea of the difficulties of embeddings. The development of syntactic rules and functions (i.e. transformation and decision procedures) to control the generation of reasonably good complex sentences will probably bring to light numerous other critical conditions usually associated with the verb structure.

Pronominalization:

Suppose now that we have already generated from C1, the sentence: JOHN SAW MARY WRESTLING WITH A BOTTLE AT THE LIQUOR BAR, as in the earlier example and wish to continue with the structure representing "John went over to help Mary with the bottle". This structure is represented by C12 and the network it dominates in Figure 4. Since the tokens JOHN, MARY, and BOTTLE have already been printed in the generation of the first sentence their subsequent use requires the substitution of pronouns, or of an anaphoric expression or an ellision.

Our method for dealing with these matters is still highly experimental and limited so far to assigning pronouns. When the token of a noun is generated as a print image, the relation pair, PRON-TOK value is put on the structure. When a subsequent generation process calls for that concept, it first checks for a PRON relation; if it is present, PRON operates as a function to return the appropriate form of pronoun for that wordsense in the surface case for the NP being generated. This approach is simple and effective for generating the instances of pronouns in the example discourse, but is obviously only a beginning in one of the more complicated areas of English semantic and syntactic structure.

Example Discourse Generations: The generator function takes a list of nodes and a grammar as its arguments. In the generation process additional new nodes may be added to the list, and nodes are removed as their generation is completed. Only verb nodes are added to the list since the generator uses verb structures as a starting point for producing a sentence. Verb structures are identified as those having form determiners and the AUX relation. In the discourse studied (see Table 5) the verb structures are heavily inter-related by having common arguments (i.e. values of semantic relations) such as MARY, JOHN, BOTTLE, etc.. This degree of inter-connection has the consequence that starting with any one verb structure, the whole discourse may be generated (or not depending on the rules for adding nodes to the list to be generated).

Examples of sequences of sentences that the system with its present grammar and syntactic functions has generated are shown in Table 8. The weaknesses of our present treatment of pronouns and anaphora is apparent in these examples. Such awkwardnesses as "a bottle cork" for "the cork" or "the cork from the bottle" are glaring. In the last example sentence, "Before John drew the cork, he went over to help Mary with a bottle", the use of "a bottle" reveals our present inability to use the information that "the bottle" is the same one with which "the cork" is associated. The data is in the semantic structure but a rule to use it has not been developed.

On the positive side, the example shows that the grammar produces reasonably good English sentences with fairly accurate choices of pronouns and time adverbs. Although the generation algorithm is still experimental and itself subject to modification, it can now support linguistic experimentation with the form and content of grammar rules and transformations for producing connected discourse.

GEN [(E1, E16), GR]

JOHN SAW MARY WRESTLING WITH A BOTTLE AT THE LIQUOR BAR.
JOHN WENT OVER TO HELP HER WITH IT BEFORE HE DREW THE CORK.
JOHN AND MARY TOGETHER DRANK THE CHAMPAGNE.

GEN [(E3, E16), GR]

MARY WAS WRESTLING WITH A BOTTLE AT THE LIQUOR BAR BEFORE JOHN HELPED HER WITH IT.
JOHN SAW MARY WRESTLING WITH A BOTTLE AT THE LIQUOR BAR.
JOHN WENT OVER TO HELP MARY WITH A BOTTLE BEFORE HE DREW A BOTTLE CORK.
JOHN AND MARY TOGETHER DRANK THE CHAMPAGNE.

GEN [(E16), GR]

JOHN AND MARY TOGETHER DRANK THE CHAMPAGNE AFTER HE SAW HER WRESTLING WITH A BOTTLE AT THE LIQUOR BAR.
JOHN BEFORE HE DREW A BOTTLE CORK WENT OVER TO HELP MARY.

GEN [(E16, E12), GR]

JOHN HELPED MARY WITH A BOTTLE AFTER HE SAW HER WRESTLING WITH IT AT THE LIQUOR BAR.
BEFORE JOHN DREW THE CORK, HE WENT OVER TO HELP MARY WITH A BOTTLE.

Table 8 Example Discourse Generations

# V Discussion and Conclusions

An important motivation of the present study was to determine what requirements on form and content of semantic nets are implied by the process of generating English sentences from them. If our claim that the semantic network adequately represents some important aspect of the meaning of discourse is to be upheld, then the very least requirement is that the nets be able to preserve enough information to allow re-generation of the sentences -- and some of their syntactic paraphrases -- from which the nets were derived. A comparison of the sentences generated in Table 7 with the sentences of the original sample discourse (p 32 ) shows that the form and content of the semantic net in Table 5 meets this minimal requirement.

In order to support sentence generation we found that our earlier definitions of semantic networks had to be augmented by providing explicit markers of the relative time-of-event for each verb structure, and that form determiner relations and values were essential for determining the surface form of the structures to be generated as sentences. We also found that numerous contextual conditions must be satisfied before embedding one sentence within another. At the syntactic level, this finding is already familiar from linguistic studies of transformational grammars, but the present study begins to show the nature of some of the semantic and stylistic conditions that must be satisfied. The most important condition we observed was that surface signals must be generated -- such as the ordering of clauses and the selection of temporal adverbs-- to reflect the underlying time-order of

conceptual events as represented in the semantic structure. Stylistic constraints against excessive depth of syntactic embedding and excessive complexity resulting from too many embeddings must also be part of the generation plan -- although there is no reason yet to think these are part of the semantic structure.

The form and content of the grammar for accomplishing the generation is taken as a question that is almost as important as the structure of the semantic nets. Many forms of generation rules could have been used, ranging from context-sensitive rewrite rules to Chomsky-type transformations. We found Woods' augmented-finite-state-network grammar most closely related in form to the semantic network structure. We modified his approach by requiring that the arcs of the grammar network name LISP functions that tested conditions and accomplished transformations on the semantic structure as it was passed from state to state in accordance with a path through the grammar net. We also required that the grammar arcs correspond in name to the semantic relations in the semantic network. This means that associated with each semantic relation there must exist a syntactic transformation function that modifies and transforms the arguments of the semantic relation toward a surface syntactic form.

We believe that this grammatical approach will prove generally satisfactory if the number of semantic relations is not too great. Semantic relations used in this system are binary relations that associate: 1) verb concepts with their nominal arguments-- such as the deep case relations suggested by Fillmore-- 2) nouns with their modifiers and determiners, 3) intersentential relators of causality, succession, implication etc.

with their sentence arguments, 4) conjunctions and their arguments and 5) concept tokens with their lexical representations. Indubitably, other forms of semantic relations also exist, but it appears reasonable to hope that the total inventory of semantic relations may be numbered in the dozens -- rathe than the hundreds. This conjecture suggests two important areas for continued research:

1) the establishment of a fairly complete inventory of semantic relations (in English and/or other languages), and

2) the definition of conditions and transformations that are required to express each relation as one or more surface structures in discourse.

An important consequence of this syntactic approach is that the grammar net shows only the ordering of the transformations, not their content. The content is in the form of a set of conditions and operations coded as LISP functions. This fact implies another area for research: the development of a convenient language for linguistic researchers to express the complex of conditions and operations that comprise each transformation function. Chomsky's transformational conventions may be suitable as such a language but we suspect that more specialized languages such as that developed by Woods (1970) offer a more convenient starting point.

Since we have not elected to use one of these languages-- favoring LISP instead-- the exact nature of the transformation functions has only been apparent to the reader where we have described them in detail as in the computational model associated with form determination. In our own defense, we are not yet convinced that many of the transformation functions we have so far developed are general enough to warrant publication.

The development of a generally true fragment of English grammar, whether in the form of transformational rules or LISP functions, is an arduous linguistic task that requires much testing and cross-checking.

The generation algorithm is a simple system that successively modifies a semantic structure by applying a series of transformation functions in an order determined by the grammar network. We believe that this algorithm is of general interest for applying a complexly ordered set of operations to a network. Further study will show whether the algorithm is also useful in applying semantic transformations for generating semantic paraphrases or for obtaining answers to questions.

Large areas of concern to the generation of natural language discourse have remained unexplored in this research. The first of these is the selection from a large semantic net of some subportion to be expressed in language. In general, this is a problem of planning a discourse or outlining an essay. In the special case of generating an answer to a question, the question-answering algorithm determines the nodes which are answers and can pass these to the generator. The general case, however, requires detailed study of the thematic organization of discourse and the invention of a discourse planning system.

Stylistic conditions on the generation of discourse concern the choice of preferred paths through the grammar network as a function of still unknown contextual and pragmatic conditions. These and conditions that dictate forms of anaphora and selection of sentence embedding transformations suggest rich areas for much additional research. In short, the universe of unexplored ideas remains far larger than the glimpses offered by this or any particular report of research.

Our glimpses concerning semantically controlled generation of English discourse are summarized in the following ten statements:

1) A grammar alone generates syntactically well-formed strings of terminal word-classes. Random lexical substitution within these classes usually results in semantically nonsensical sentences.

2) A semantic network and its lexicon alone generates semantically understandable strings of words that are not syntactically well-formed sentences.

3) Semantically controlled generation requires that each semantic relation correspond to one or more syntactic relations and to the associated transformations that can change a portion of the semantic representation toward a surface syntactic form. The choice of rules or paths through the grammar is accomplished by selecting only rules whose elements correspond to semantic relations in the net that is being worked on.

4) The grammar imposes an ordering on the application of the transformations. Once a particular rule is applied the grammar alone limits further choices to a non-contradictory ordered set of paths. Within this set, the paths to be chosen are further restricted to those that can represent the semantic relations in the network under consideration.

5) Generation is begun by selecting a node from the net, labelling it S, and entering the grammar at the rule or path labelled S.

6) The first step in the generation process appears to be the selection of a set of form-determiners in terms of values for Voice, Form, Aspect, Tense and Mood. This set probably also includes the choice of modals as has been suggested by Fillmore's (1968) rule: S - Modality + Proposition.

7) At various points in the generation process inverse semantic relations such as AGT*, LOC*, etc. are encountered. These offer the possibility of embedding a sentence. The choice of embedding or not at that point is made by complex criteria which so far concern relations between the head verb of the sentence and the verb of the sentence to be embedded. The result of applying such criteria probably also affects the values for form determiners, although we have not yet seen how.

8) These observations with regard to embedding and form determination suggest that there exist criteria still unknown for selecting various forms of sentence as a function of context. Applying decision rules based on such criteria would appear to result in a selection of values for the form determiners.

9) This further suggests that a discourse generator stands above a sentence generator in order to select portions of the semantic net from which sentences are to be generated, and to exert thematic control via choice of subjects, form determination, and the ordering of sentences.

10) The generation of anaphoric references is also apparently subject to complex decision rules sensitive to the context. These must account for the conditions that lead to a choice of a pronoun or some shortened nominal expression, as well as accounting for an appropriate use of determiners in many cases. So far, rules have been developed only for generating pronouns respecting gender, number and case. A pronoun is substituted for any reference to a noun that has already been used in generating a sentence of the discourse.

The system is available as a LISP 1.5 program for the CDC 6600. Only minor modifications in the program are required to fit it to most other LISP implementations. The program so far is expressed in about a hundred lines of LISP expressions. It is available on request from the authors.

# REFERENCES

1. BACH, E. and HARMS, R. T. "Nouns and Noun Phrases." IN Universals in Linguistic Theory. Holt, Rinehart and Winston, Inc., Chicago, 1968.

2. CARBONNEL, J.R. "Mixed-Initiative Man-Computer Instructional Dialogues." BBN Report #1971, Bolt, Beranek and Newman, Inc., May, 1970.

3. COLBY, K.M., TESLER, L. and ENEA, H. "Experiments with a search algorithm for the data base of a human belief structure." Proc. Int. Jt. Conf. Art. Intel., Washington, D. C., 1979, pp. 649-654.

4. COLES, S. L. "An on-line Question-Answering System with Natural Language and Pictorial Input." Proc. ACM 23rd Nat. Conf.1968, Brandon Systems Press, Princeton, N.J., pp. 157-167.

5. FILLMORE, C.J. "Types of Lexical Information." Ohio State Univ., Computer and Info. Sci. Res. Ctr. Working Papers in Linguistics #2, Columbus, Ohio, 1968.

6. FRIEDMAN, J. "A Computer System for Transformational Grammar." Comm. ACM 12, 6(June 1969), 341-348.

7. GREEN, C. C. and RAPHAEL, B. "Research on Intelligent Question-Answering Systems." Proc. ACM 23rd Nat. Conf., 1968, Brandon Systems Press, Princeton , N.J., pp. 169-181.

8. JACOBS, R. A. and ROSENBAUM, P.S. English Transformational Grammar. Blaisdell Publ. Co., Waltham, Mass., 1968.

9. KATZ, J.J. "Recent Issues in Semantic Theory." Foundations of Language 3 (1967), 124-194.

10. KELLOGG, C. H. "A Natural Language Compiler for On-line Data Management." AFIPS Conference Proceedings, Vol. 33, Proc. AFIPS 1968 Fall Joint Comput. Conf. Vol.33, Thompson Book Co., Washington, D.C., pp. 473-493.

11. LAKOFF, G. "Generative Semantics." IN Semantics - An Interdisciplinary Reader in Philosophy, Linguistics, Anthropology and Psychology. London: Cambridge Univ. Pr., 1969.

12. LAKOFF, G. and ROSS, J. "Is Deep Structure Necessary." Publications of Indiana Univ. Ling. Club - Preprint, 1969.

13. LEECH, G. N. Towards a Semantic Description of English. Univ. Press,Bloomington, Ind., 1970.

14. McCAWLEY, J. D. "The Role of Semantics In a Grammar." IN BACH, E., and HARMS, R.T. Universals in Linguistic Theory. Holt, Rinehart and Winston, Inc., Chicago, 1968.

15. QUILLIAN, M. R. "Semantic Memory." Ph.D. Th., Carnegie-Mellon Univ., Pittsburgh, Pa., Feb., 1966.

16. QUILLIAN, M.R. "The Teachable Language Comprehender." Comm.ACM 12, Bolt, Baranek and Newman, Cambridge, Mass., Jan. 1969, in preparation.

17. SIMMONS, R. F., BURGER, J.F., and SCHWARCZ, R.M. "A Computational Model of Verbal Understanding." Proc. AFIPS 1968 Fall Joint Comput. Conf., Vol. 33, Thompson Book Co., Washington, D.C., pp. 441-456.

18. SIMMONS, R. F. "Natural Language Question-Answering Systems: 1969." Comm. ACM, Vol. 13 #1, Jan. 1970, pp. 15-30.

19. SIMMONS, R. F. "Some Semantic Structures for Representing English Meanings." Univ. Tex., Austin, Comp. Sci. Dept., Preprint, Nov. 1970.

20. SIMMONS, R. F. "Some Relations between Predicate Calculus and Semantic Net Representations of Discourse." Univ. Texas, Austin, Comp. Sci. Dept., Preprint, Nov. 1970.

21. SIMMONS, R. F. "Compiling Semantic Networks from English Sentences." (In preparation).

22. WOODS, W. A. "Augmented Transition Networks for Natural Language Analysis." Aiken Comp. Lab., Harvard Univ., Cambridge, Mass., December, 1969.

23. WOODS, W. A. "Transition Network Grammars for Natural Language Analysis." Comm. ACM, Vol. 13, #10, October, 1970.