# QUESTION ANSWERING VIA CANONICAL VERBS AND SEMANTIC MODELS: GENERATING ENGLISH FROM THE MODEL

Jonathan Slocum

January 1973

Technical Report NL 13

Natural Language Research for CAI

Supported by

The National Science Foundation

Grant GJ 509X

The Department of Computer Sciences

and CAI Laboratory

The University of Texas

Austin, Texas 78712

## ABSTRACT

A system is described for generating English from a semantic net model of the world composed of nodes which represent objects, events, or properties of objects or events, and directed labelled arcs which represent static relationships among the objects, events and properties. The generative grammar is an Augmented Finite State Transition Network which "parses" syntactic patterns retrieved from the lexicon and produces as a side effect English expressions. Some theoretical and practical considerations are advanced.

The system is in use as part of a natural language question-answering system programmed in GROPE. Preliminary findings involve methods for determining the surface structure of the expressions (sentences and noun phrases) to be generated.

## PREFACE

This document is the third in a group of three which discuss aspects of a particular natural language question-answering machine. The device is programmed as three independent modules -- of which the third is our concern.

(1) The parser takes as input an English sentence and produces as output a primitive (list) form of a semantic net (Thompson, 11).

(2) The data base manager maps this net onto the network model of the world, and produces as output some node -- perhaps from the data base -- to be generated as a response (Hendrix 2).

(3) The generator operates on this node to produce natural language output -- which may vary from a short "pat" response such as OK, or I-DO-NOT-UNDERSTAND, to complete English sentences -- whereupon the cycle is repeated.

For a more complete description of the first two modules, the reader is referred to the papers by Thompson and Hendrix.

# QUESTION ANSWERING VIA CANONICAL VERBS AND SEMANTIC MODELS: GENERATING ENGLISH FROM THE MODEL

## Introduction

One important problem confronting the natural language question-answering system designer is that of presenting answers in some form easily comprehended by the user. For text-based systems such as we are discussing, natural language would seem to be the most appropriate medium, if not the simplest to implement. Several systems have attacked this response-mode problem with forms of English generators -- notably those of Simmons, Burger and Schwarcz (8), Quillian (4), Carbonnel (1), Simmons and Slocum (9), and Winograd (12). Generally speaking, the approaches have all involved ordered transformations upon the "deep" semantic net structures to derive surface structures more-or-less resembling English. Concerning the discourse generation system of Simmons and Slocum in particular, two main points emerge with respect to single-sentence generation: (1) the grammar comprised _one_ set of (surface pattern) rules which was intended to serve all classes of surface structures, and (2) the transformations modified (destroyed) the data base, so that the maintenance of a copy was necessary. This paper will indicate our solutions to these problems in the context of our introduction to recent work by Woods (13) and (14), Schoene and Celce (6), and Schank (5).

## Background

The reports by Schoene, Celce and Schank showed our earlier notion of a generative grammar (9) to be weak in that it could not take advantage of the patterns of case relationships peculiar to particular verb types. That is, a grammar should allow branching in some form, contingent upon the choice of surface verb, rather than just the arguments present. Furthermore, given the presence of, say, three (predicate) arguments, our form of grammar allowed exactly one ordering of those arguments in the surface structure, whereas English allows many -- up to six in this example:

> The papers were blown by the wind from the pier to the road.
> The papers were blown by the wind to the road from the pier.
> The papers were blown to the road from the pier by the wind.
> The papers were blown from the pier to the road by the wind.
>
> .
> .
> .

There are three PPs which are being generated, but they may appear in six differerent orders -- although some persons might argue the acceptability of a few of the six. Nevertheless, the fact remains that a generative grammar must allow for these possibilities; our previous methods did not do so.

Roughly coincidental with this "revelation" was our first implementation of Woods' Augmented Finite State Transition Network (AFSTN) automaton. This system allows a grammar to have variables (registers) which can be set, tested, and reset by certain operations; in essence a "Woods grammar" is a program in graph form. Woods' original system was oriented toward parsing (English) into tree-structures -- as evidenced by his arc types and other operations -- and the functions he admitted were somewhat restrictive. We first generalized the system to allow allow any function available in the host language (LISP), and as a result induced a sizeable increase in its flexibility. In a later (GROPE) implementation (Matuszek and Slocum, 3), certain of the tree-building operations were considered non-essential and hence discarded.

Our current generator is written as an AFSTN grammar. The use of grammar-program registers to contain temporary values has enabled us to generate English without modifying the data base: our earlier system (9) stored temporary values in the data base. The lexicon maintains (in list form) the argument ordering possibilities allowed for each surface verb. In terms of the AFSTN system, an ordering (herein called a syntactic pattern, or rule) is a "sentence" to be "parsed". That is, both natural language sentences and these patterns may be viewed as control strings for the parsing and generative grammars, respectively, which (grammars) produce as their side-effects semantic nets or English sentences.

We should pause here for a moment and define the concept "surface verb." Simply stated, a surface verb is (the root form of) any verb one normally utters in the course of a natural language expression. This terminology is to distinguish such forms from what we call "canonical verbs". Schank (5) and others have posited the existence of "deep" verbs, which unify in the deep structure the meaning common to possibly many of the verbs with which we are ordinarily acquainted. For instance, the verbs BUY, SELL, COST and PAY (among others) are different (surface) ways of expressing what is essentially one concept: a transfer or exchange of ownership of (two) items by (two) persons. Therefore, sentences with various surface verbs will be parsed into semantic nets appropriate to their canonical base, and sentences generated from such a base might employ any of the surface verbs allowed by the canonical event name. (Thus if we forget question-answering for the moment, we see that we have a device capable of producing all legal syntactic and semantic paraphrases -- excepting noun synonyms -- of an input sentence.) In our system, the particular verbs mentioned above are unified in the data base under the canonical verb EXCH (for EXCHANGE), and do not themselves appear anywhere in the model of the world.

## Sentence Generation

Once the lexicon assumes the burden of defining the allowable syntactic patterns for each surface verb, the generative grammar writer is free to concentrate on (1) the over-all grammar control structure, (2) the noun (and prepositional) phrase generation problems, and (3) the verb-string generation problem. The major function of (1) is to cause the proper execution of (2) and (3), then to concatenate the partial results and return the newly-generated S. These are relatively straightforward matters, especially when one ignores (disallows) embedded sentences and full discourse generation. (In view of the fact that this is intended to be a simple answer-generating device, rather than a truly conversational machine, these restrictions are not unwarranted.) But there exists the problem of choosing the appropriate syntactic pattern.

The sample semantic net based on the canonical verb EXCH (see Figure 1) may be expressed using the surface verbs BUY, SELL, COST, etc. Depending on the particular surface verb chosen, some rule (see Table I) may be chosen as the "control string," or input sentence, for the generative

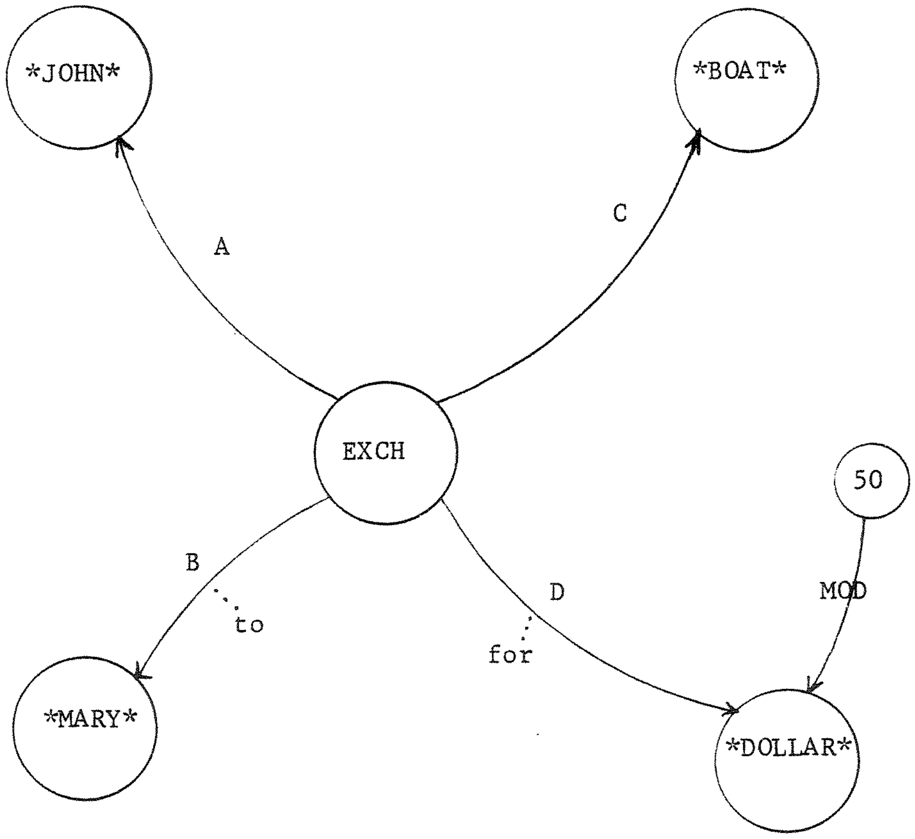

FIGURE 1

A sample semantic (data base) net produced as a consequence of the input sentence:

"John sold the boat to Mary for fifty dollars."

| | | | | | |
|---|---|---|---|---|---|
| BUY | V | L00001 | PAY | V | L00003 |
| BUYS | V | L00001 | PAYS | V | L00003 |
| BOUGHT | V | L00001 | PAID | V | L00003 |
| BUYING | V | L00001 | PAYING | V | L00003 |
| COST | V | L00002 | SELL | V | L00004 |
| COSTS | V | L00002 | SELLS | V | L00004 |
| COSTING | V | L00002 | SOLD | V | L00004 |
| | | | SELLING | V | L00004 |

| | | |
|---|---|---|
| EXCH | VERBS | (L00001 L00002 L00004 L00003) |
| L00001 | RULES | ( [ B VACT C (from A) (for D) ] [ C VPAS (from A (for D) ] ) |
| | CANON | EXCH |
| | INF | BUY |
| | SG3 | BUYS |
| | PST | BOUGHT |
| | -EN | BOUGHT |
| | -ING | BUYING |
| L00002 | RULES | ( [ C VACT (B) D ] ) |
| | CANON | EXCH |
| | INF | COST |
| | SG3 | COSTS |
| | PST | COST |
| | -EN | COST |
| | -ING | COSTING |
| L00003 | RULES | ( [ B VACT (A) (D) (for C) ] ) |
| | CANON | EXCH |
| | INF | PAY |
| | SG3 | PAYS |
| | PST | PAID |
| | -EN | PAID |
| | -ING | PAYING |
| L00004 | RULES | ( [ A VACT C (to B) (for C) ] [ A VACT B C (for D) ] [ C VPAS (to B) (by A) (for D) ] ) |
| | CANON | EXCH |
| | INF | SELL |
| | SG3 | SELLS |
| | PST | SOLD |
| | -EN | SOLD |
| | -ING | SELLING |

TABLE I

Some lexical structure with optional rule elements denoted by enclosure within parentheses.

grammar. In any desired fashion (say, by random choice) one may choose a surface verb -- we will choose L00001 as our first example. We now have a choice of two patterns for our verb -- we shall choose the rule [ B VACT C (from A) (for D)] . (For the moment we shall not concern ourselves with the mechanics of NP, PP, and Verb String generation, but instead concentrate on flow-of-control.) The first element in the rule, B, indicates that an NP is to be generated (as what is commonly called the <u>subject</u> of the sentence) from the node satisfying case relationship B of the node labelled EXCH in Figure 1: *MARY*. This NP generation produces the result MARY. The second element in the rule, VACT, indicates the <u>voice</u> (here, active) in which the sentence is to be generated. The Verb String generation produces (for instance) BOUGHT. The next (third) element in the rule, C, indicates that an NP is to be generated from the node satisfying case relationship C: *BOAT*. The result might be THE BOAT. The next element, (from A), indicates that the node *JOHN* is to be generated as a PP, using the preposition <u>from</u> -- resulting in FROM JOHN. The last element in the rule, (for D), indicates that the node *DOLLAR* is to be generated as a PP, using the preposition <u>for</u> -- resulting in FOR 50 DOLLARS. Since the entire rule has now been "parsed," the final function of the grammar is to string-together all these intermediate results and return the sentence:

MARY BOUGHT THE BOAT FROM JOHN FOR 50 DOLLARS.

Without going into detail, we can see that the choice of PAY and the rule [ B VACT (A) (D) (for C) ], when applied to the identical deep structure, would result in the output sentence:

MARY PAID JOHN 50 DOLLARS FOR THE BOAT.

It is worth pointing out here that the choices COST and [C VACT (B) D] would produce the sentence:

THE BOAT COST MARY 50 DOLLARS

This obviously contains less information than the underlying structure as seen in Figure 1, but note the verb COST does not allow us to include the NP JOHN, or the PP FROM JOHN, etc. Thus we are justified in associating these syntactic patterns with surface verbs (as shown in Table I) rather than with their canonical form (EXCH). Also, non-generation of known material is shown to be possible -- even necessary, in some cases.

Now consider the problem of generating a sentence from an incomplete structure, as in Figure 2. If generation is attempted with the same verb and rule choice as in the last example, a non-sentence would be returned:

THE BOAT COST MARY

Thus the choice of surface verbs and patterns cannot be truly random: it is necessary that some mechanism test a tentative pattern against the data base net to insure that the required arguments (case relationships) are present in the net. In this example, the last rule element -- D -- which is <u>not</u> optional, is missing from the net. The testing mechanism must be responsible for rejecting bad choices, so that another choice can be made. Note that any of the other rules in Table I would be acceptable, since in all these cases the presence of argument D in the output string is defined to be optional.

It is possible that one might not wish the system to consistently generate the maximally informative sentence allowed by a rule, even though all elements be present in the data base. For instance, since several of the rules indicate the optionality of case relations A and D, one might

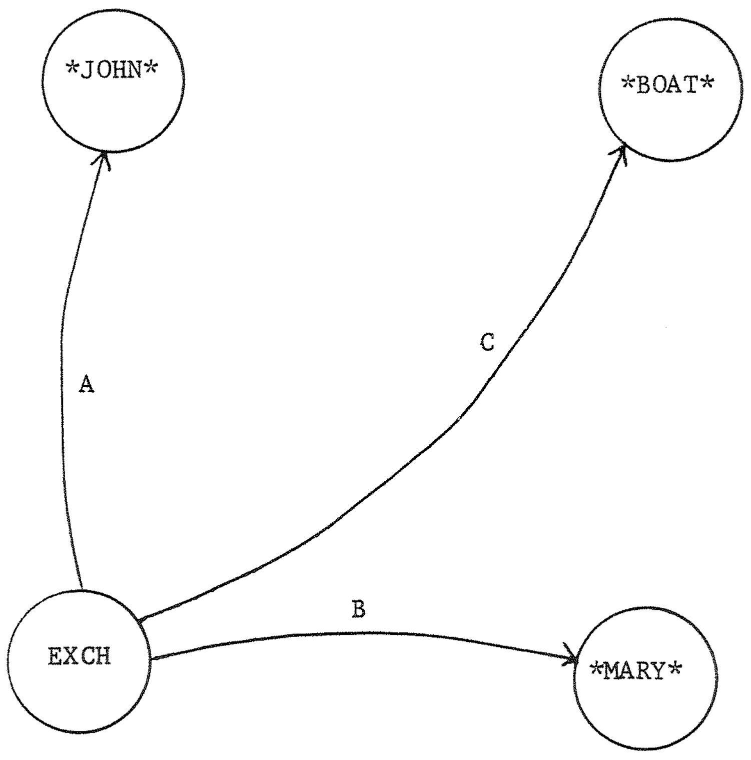

FIGURE 2

A sample net produced as the consequence of the input:

"John sold Mary the boat."

wish to allow their deletion from the surface structures produced -- or, better yet, allow their non-generation. A sentence example (from Figure 1) is:

MARY BOUGHT THE BOAT FOR 50 DOLLARS

This "deletion" could be handled through explicit storage of all of the variants of a rule, with some optional element(s) deleted from each rule; however, this is unnecessarily redundant. (The rule of this type corresponding to the above example is [B VACT C (for D) ].) Instead, the grammar itself may be constructed so as to allow for this possibility -- perhaps by random omission of optional NP or PP generations, or by any other heuristic which the grammar writer may wish to employ. The problem to be recognized here is that one would prefer not to allow the possibility of generating a response to a question, in which the desired information (the answer) has been "optionally deleted." But thereis an additional possibility for answer generation which solves this problem: "answer-only" generation.

## Noun Phrase Generation

So far we have been concerned with the generation of necessarily whole sentences. However, in the particular case of humans, probably most (spoken) answers are not sentences, but rather (noun) phrases. There would seem to be no reason why a mechanical answer generator must be constrained to the production of "complete" sentences. The AFSTN system allows one to pass initial control to any node in the grammar: this facility is employed in our answering system, as we sometimes choose to generate an NP rather than an S.

Consider Figure 1. The simplest answer to the question "Who sold the boat?" is the NP "John." "Mary" is the answer to the question "Who bought the boat?" Now if the output node from stage (2) of our system is an <u>event</u> node (for example, EXCH), then the generator should produce a sentence. But most often the answer is an <u>entity</u> node (such as *JOHN*), in which case an NP is to be generated. Since the generation of JOHN from the node labelled *JOHN* would not particularly clarify the problems in NP generation, we shall consider the example network in Figure 3 and see how an NP is produced in this "worst case."

The nodes labelled *MAN*, *WAGON* and *GIRL* (Figure 3) are <u>entity</u> nodes: they correspond on a one-to-one basis with particular objects (or in the case of *WAGON*, a particular <u>set</u> of objects) in the real world. OLD, 15, RICKETY, RED, and LITTLE are <u>predicates</u> about the entity (set) *WAGON*; thus this entity has the attributes AGE, NUMBER, CONDITION, COLOR and SIZE, whose values are the respective predicates. While all of these predicates may be thought of as MODifiers, we shall see that there is a good reason for being more precise. Upon inspection of Figure 3, we note that these propositions look much like event nodes -- they both have directed, labelled arcs which point to entity nodes. These labels might even be considered to be "case relationships." But there is one important distinction: <u>events</u> are by nature <u>one</u>-time objects -- they happen, then they are over; a proposition, on the other hand, is static -- it goes on and on, until something (an event) happens which changes its "truth value."

If one were to randomly generate the modifiers of *WAGON* in the course of generating that node as an NP, the result might be:

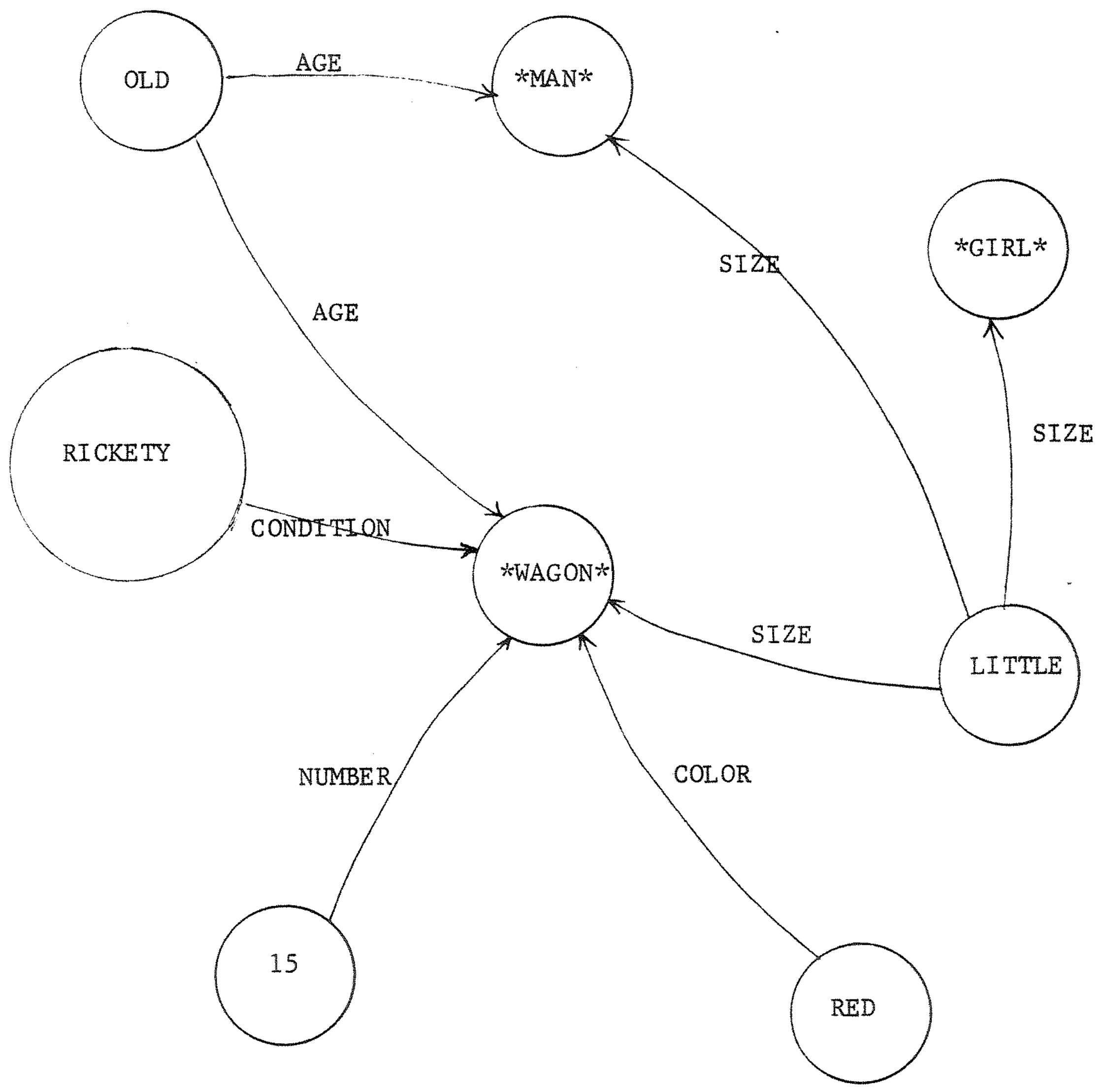

FIGURE 3

A net encompassing three entities and five properties

THE OLD RICKETY 15 RED LITTLE WAGONS

or, THE RED LITTLE 15 RICKETY OLD WAGONS

or, THE 15 LITTLE OLD RICKETY RED WAGONS

We recognize only the last as being acceptable. What makes it different from the others, obviously, is the ordering of the modifiers; but if all modifiers were only MODifiers -- as in our earlier system -- how could one determine the surface ordering? This is our justification for the classification of all modifiers into categories, and the ordering of the categories to control the ordering of the surface modifiers. (Winograd (12) also ordered his modifiers.) We loosely refer to these categories as attributes. Now we might posit another "control string" which is used when generating NPs. The acceptable (3rd) example above would indicate that a proper control string is: NUMBER SIZE AGE CONDITION COLOR . (No claim is being made for the completeness of this example -- rather, the system designer establishes his own categories through his particular vocabulary subset, backed-up with a suitable NP grammar.) Now the NP grammar has the relatively simple task of generating the modifiers as the control string leads it to find them, then generating the surface noun in the singular or plural as indicated by the NUMBER modifier. (Naturally, if the value of NUMBER is S or PL -- rather than a numeral -- it is not included in the surface string.) For simplicity, proper nouns and certain others (like "dollar") do not take determiners; other nouns take the determiner "the" by default.

As one example, the control string above orders the generation of NUMBER (15), SIZE (little), AGE (old), CONDITION (rickety), and COLOR (red), to produce the adjective string:

15 LITTLE OLD RICKETY RED

When this is preceded by the determiner "the" and followed by the (plural, due to _15_) noun "wagons," the resultant NP is:

THE 15 LITTLE OLD RICKETY RED WAGONS

Similarly, and since _all_ categories are optional by fiat, this same control string would cause the grammar to produce the ordered adjectives "little" (SIZE) and "old" (AGE), in the course of generating

THE LITTLE OLD MAN

from the node *MAN* in Figure 3.

As a result of this discussion we have seen that the necessary control structure for an NP generation grammar is relatively simple and that it resembles the top-level control structure; furthermore it may be noted that, due to the properties of the AFSTN system, it is not necessary for the NP grammar subset to "know" whether it is generating an NP in isolation, or in the context of a complete sentence. In either case it merely produces the surface string NP and "POPs it up" to the next higher level. In the case of the isolated NP this POPping will exit the grammar, returning the string to the function which called for the generation of the NP, while in the case of complete sentence generation the POP will return the string to the place in the grammar where the PUSH NP was executed -- to be incorporated into the overall S being generated.

## Verb String Generation*

One might expect that there is a "control string" for guiding VS generation -- and there is: [VOICE FORM ASPECT·TENSE CASE MOOD ]. Table II indicates the possible values for each of these properties, and the action to be taken for each value. Not all combinations of values are allowed in English: when this "form determination" algorithm is entered, it is assumed that some mechanism (a still unknown "conversational controller") has indicated the desired values for each of these properties. It is this mechanism that must insure the values form a legal combination.

The actions taken by the algorithm involve pushing a new (FIRST) surface form onto the string, popping the FIRST element (the last one pushed) and remembering it, and (by means of a pop and push) changing the FIRST element to some other surface form. The lexicon may of course be freely consulted during this process. Initially, the string is composed of the root form of the (main) surface verb, in FIRST position. In our example (see Table II) the PASSIVE transformation changes "SELL" into its past participle form and adds "BE"; the PERFECT transformation changes "BE" to <u>its</u> past participle form and adds "HAVE"; the FUTURE transformation adds "WILL"; and the NEGATIVE transformation pops "WILL", adds "NOT", then replaces "WILL", resulting in the verb string: "WILL NOT HAVE BEEN SOLD."

---

* Simmons and Slocum (9) advanced a VS generation algorithm; our only modifications to that algorithm are (1) the VOICE section is no longer responsible for setting-up SUBJECT and OBJECT values -- since these are explicit in our syntactic patterns, and (2) the negation and emphatic transformations are introduced. However, for completeness' sake, we include a short tabular description of that algorithm and a clarifying example.

| Property | Possible Values | Action | Example Value | Example String |
|---|---|---|---|---|
| | | enter | | SELL |
| VOICE | ACTIVE | none | | |
| | PASSIVE | change FIRST to -EN form | x | SOLD |
| | | push "BE" | | BE SOLD |
| FORM | (simple) | none | x | |
| | PROGRESSIVE | change FIRST to -ING form | | |
| | | push "BE" | | |
| | EMPHATIC | push "DO" | | |
| ASPECT | (simple) | none | | |
| | PERFECT | change FIRST to -EN form | x | BEEN SOLD |
| | | push "HAVE" | | HAVE BEEN SOLD |
| TENSE | PRESENT | change FIRST to agree in person and number with the SUBJECT | | |
| | PAST | change FIRST to PAST form (insure person and number agreement if FIRST was "BE") | | |
| | FUTURE | push "WILL" | x | WILL HAVE BEEN SOLD |
| CASE | AFFIRMATIVE | none | | |
| | NEGATIVE | pop FIRST | x | HAVE BEEN SOLD |
| | | push "NOT" | | NOT HAVE BEEN SOLD |
| | | push (old) FIRST | | WILL NOT HAVE BEEN SOLD |
| MOOD | INDICATIVE | none | x | |
| | INTERROGATIVE | pop FIRST and <u>front</u> it (place before already-generated SUBJECT) | | |
| | | exit | | |

TABLE II

The verb string generation algorithm

## Conclusions

The approach taken shows generation of some English expressions (answers) to be a relatively simple task. The ÀFSTN system has all the power necessary, while at the same time incorporating a control structure which greatly reduces the complexity of the demands on the grammar writer. The overall structure of simple (as opposed to compound) English sentences can be modelled with syntactic patterns associated (via the lexicon) with the surface verbs in the vocabulary. Similarly, noun phrases are constrained with respect to the ordering of their adjectives (if any) through the use of a single rule. Prepositional phrases are obviously just (appropriate) prepositions concatenated to NPs: the rules and/or data base conspire to determine the "appropriate" prepositions. Finally, there is also a control string suitable for guiding the generation of the verb (and its auxiliaries) provided the appropriate event node has attributes (TENSE, ASPECT, etc.) whose values will key the generation in an acceptable manner. We have intentionally avoided saying anything about the nature of the "conversational controller" which can cause this device to be fully conversant, since we have not yet followed-up this advance with research along those lines.

## REFERENCES

1. Carbonnel, J. R., "AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction," In IEEE Transactions on Man-Machine Systems, MMS-11, #4, December 1970.

2. Hendrix, G. G., "Question Answering Via Canonical Verbs and Semantic Models: A Model of Textual Meaning," The University of Texas, Austin, Technical Report NL 12, January 1973.

3. Matuszek, D. and Slocum, J., "An Implementation of the Augmented Transition Network System of Woods," The University of Texas, Austin, Technical Report NL9, October 1972.

4. Quillian, M. R., "The Teachable Language Comprehender," Comm. ACM, vol. 12, #8, August 1969, pp. 459-476.

5. Schank, R. C., Conceptual Dependency: A Theory of Natural Language Understanding, manuscript, Stanford University, May 1971.

6. Schoene, W. J. and Celce, M., Research on Demonstration of Natural Language Computer-Assisted Instruction, Draft of Final Report, System Development Corp., Santa Monica, California.

7. Simmons, R. F., "Semantic Networks: Their Computation and Use for Understanding English Sentences," The University of Texas, Austin, Technical Report NL 6, May 1972.

8. Simmons, R. F., Burger, J.F., and Schwarcz, R. M., "A Computational Model of Verbal Understanding," Proc. AFIPS 1968 FJCC, vol. 33, AFIPS Press, Montvale, N. J., pp. 441-456.

9. Simmons, R. F., and Slocum, J., "Generating English Discourse from Semantic Networks," Comm. ACM, vol. 15, #10, October 1972, pp. 891-905.

10. Slocum, J. and Friedman, D. P., GROPE: A GRaph OPErations Extension of Fortran. In preparation, The University of Texas at Austin, 1973.

11. Thompson, C. W., "Question Answering Via Canonical Verbs and Semantic Models: Parsing to Canonical Verb Forms," The University of Texas, Austin, Technical Report NL 11, January 1973.

12. Winograd, T., Understanding Natural Language, Academic Press, New York, 1972.

13. Woods, W. A., Augmented Transition Networks for Natural Language Analysis. Aiken Comp. Lab., Harvard University, Cambridge, Mass. December 1969.

14. Woods, W. A., "Transition Network Grammars for Natural Language Analysis," Comm. ACM, vol. 13, #10, October 1970, pp. 591-606.