A Computational Approach to Grammatical Coding of English Words*

SHELDON KLEIN AND ROBERT F. SIMMONS

System Development Corporation, Santa Monica, California

Abstract. As a first step in many computer language processing systems, each word in a natural language sentence must be coded as to its form-class or part of speech. This paper describes a computational grammar coder which has been completely programmed and is operational on the IBM 7090. It is part of a complete syntactic analysis system for which it accomplishes word-class coding, using a computational approach rather than the usual method of dictionary lookup. The resulting system is completely contained in less than 14,000 computer words. It processes running English text on the IBM 7090 at a rate of more than 1250 words per minute. Since the system is not dependent on large dictionaries, it operates on any ordinary English text. In preliminary experiments with scientific text, the system correctly and unambiguously coded over 90 percent of the words in two samples of scientific writing. A fair proportion of the remaining ambiguity can be removed at higher levels of syntactic analysis, but the problem of structural ambiguity in natural languages is seen to be a critical one in the development of practical language processing systems.

1. *Introduction*

1.1. *Introduction to the System.* The purpose of this paper is to present a system for the mechanical coding of English words according to grammatical classes. The computational grammar coder described is the first component in a syntactic analysis program which is part of a larger question-answering system called protosynthex [8]. The total parsing system referred to in [8] performs a phrase structure analysis of English. This system is checked out on the IBM 7090 and is being translated onto the Philco 2000 and the AN/FSQ-32.

It will be seen that the use of this computational grammar coder (hereafter referred to as CGC) can be considered as an alternative to the use of a very large dictionary and that it can also serve as a context analyzer which eliminates many ambiguities of word classes arising from the consideration of words in isolation.

The usual automated method for obtaining word grammar codes (parts of speech) for vocabularies as large as those in ordinary scientific text has been to use tables in the form of dictionaries containing word code information for 25,000 to 75,000 words. Large dictionaries of this type are actual or anticipated components in the natural language processing systems of such researchers as Zellig Harris [3], Sydney Lamb [4], Anthony Oettinger [6] and Victor Yngve [9].[1] Robert Lindsay's SAD SAM system uses the dictionary approach but with vocabulary limited to the 800 words of basic English [5]. (It should be noted here that the CGC system does not rely on the method of transformations.)

The purpose of this paper is to describe and illustrate an alternate approach, the computation of grammar codes. In addition to furnishing results theoretically

* Received March, 1962; revised November, 1962.

[1] One outstanding exception is the work of G. Salton and R. W. Thorpe [7].

interesting as an exercise in morphotactics, the primary advantage of computation is that it avoids the labor of constructing a very large dictionary and permits a system to encode words it has never before encountered.

The CGC actually uses a mixed approach. It does use some dictionaries and tables. These, however, contain a relatively small number of English words (total under 2000). In addition, they contain grammar environment recognition data in the form of structural formulas. The system can encode words *not* in its dictionaries because English, like all natural languages, is highly structured. Given a small amount of grammar code information plus the ability to recognize significant contextual features, the CGC program can deduce additional structure.

Compactness is another important feature of the CGC system. It makes use of approximately 10,000 IBM 7090 machine words for its tables and 3300 machine words for its running program.[2] Accordingly, it operates entirely in 7090 core storage (32K) and leaves half of core available for other systems. The CGC tags words in an English text (e.g. *Encyclopedia Americana*) at a rate of more than 1250 words per minute. The parsing system in which it is used promises to run much faster than systems using dictionaries that must be stored on tapes.

Also of importance is the fact that the grammatical analysis made by the CGC can be changed without modifying the running program, since most changes need be made only in dictionary and table entries. This modification feature suggests that the CGC could be used in a variety of English language processing systems.

1.2 *General Remarks about Grammar.* Knowledge about the grammatical structure of a written English text is usually essential for its manipulation in an information-processing system. The pre-storage of grammatical information by tagging elements in a text can be viewed as encoding. The extent of such grammatical coding is dependent upon the needs of particular information-processing systems. But there may be a necessary minimum of grammatical analysis required for any language-processing system, including those used in information retrieval and machine translation. Without grammatical analysis, the number of programmed rules necessary for processing textual material might become unmanageable. However, the number of rules can be kept relatively small because languages have structures.

A given body of written language text is not a random collection of symbols. The forms present, whether they be letters or graphemes, words or morphemes, co-occur in a restricted set of arrangements. Consider the English sentence *He might ..... been going.* The restrictions of individual word co-occurrence permit only the appearance of *have* in the blank position. Other distribution restrictions limit the substitution in blanked-out positions to a large number of words, as in *The ..... is on the table.* and *John ..... a fish.*

Distribution restrictions can define classes of words. The sets of single words which might occur in the last two example sentences are members of classes usually labelled NOUN and VERB. Other grammatical classes such as ADJECTIVE, ADVERB, etc., can be similarly analyzed.

Extending the size of substitution items to include permissible occurrences of

[2] The CGC was written using JOVIAL, an ALGOL type language.

strings of words permits the definition of more complex classes. For example, *it*, *The book*, and *The big red book* appear as members of the same distribution class in such contexts as ..... *is on the table.*

After identifying the class membership of the individual words in phrases it is possible to determine phrase-distribution classes in terms of these class units. Thus, one mode of analysis might decide that each of the class sequences, NOUN, ARTICLE NOUN, and ARTICLE ADJECTIVE NOUN, are members of a more complex class called NOUN PHRASE. Additional analysis could yield other phrase-type classifications.

The analysis of phrase classes permits the discovery of additional facts of co-occurrence restriction in terms of the phrase units themselves. Additional classes of a more complex type, the units of which are permissible strings of phrases, can then be defined. These classes might be labelled SUBJECT OR PREDICATE.

Many linguists view these classes as a hierarchy of levels in which the distributionally defined classes of one level form the membership units of the distribution classes of the next higher level.

Tagging sentences in a computer-stored written English text with grammatical information permits a variety of data manipulations with a minimum of programmed rules. For instance, some of the rules necessary for translating a sentence such as *The red book is on the table* into another language need be formulated only in terms of general classes, e.g. subject and predicate; noun phrase, verb phrase, verb modifying phrase; adjective, article, noun, verb, and preposition.

2. *Grammar Codes and Computational Tests*

For the purpose of the grammar coder, a word is defined as that which occurs between two blanks in a written text. Most punctuation marks are treated as words.

The number of grammatical classes in English depends on one's analysis. Ultimately, each word belongs to its own unique class. The CGC recognizes 30 classes of words (see Table 1). The system is designed to permit the recognition of several hundred word classes with only trivial modification of table format accomplished with little programming effort. Class membership is assigned on the basis of form, of structural function, and/or of distribution. The names of the classes are arbitrary.

For the purpose of easy communication the class names have been made similar to many used in conventional normative grammars. A label such as NOUN is only mnemonic. The CGC may tag words at certain times in such a fashion as to appear to contradict the grammar rules the reader may have learned from conventional grammars. Because the analysis may be one of function, a form normally considered an adjective, for example, will occasionally be tagged NOUN because it functions as a noun; for example, *red* is tagged NOUN in *He chooses the red.* A word is classified solely on the basis of function only when the Context Frame Test (Section 2.6) is the only test yielding information about

TABLE 1. WORD CLASSES

Label used in Computer Output	*Full Name*	*Examples and Comments*
ADJ	Adjective	The customary usage of the term: noun modifiers including *certain, red, careful,* etc. Also included temporarily are quantifiers: numbers and words such as *many, some, no.*
ADV	Adverb	Words ending in *-ly* plus such forms as *never, too, also, likewise,* etc.
NOUN	Noun	The customary usage of the term; also occasional functional equivalents as *red* in *He chose the red.*
VERB	Verb	All verbs except the types classified in the following. The distinction between transitive and intransitive is made on the basis of syntactic context by the proto-synthex grammar machine, which utilizes the CGC.
VERB IS	The verb *to be*	*is, was, be,* etc.
AXV	Auxiliary verb	All auxiliary verbs that may not also function as verbs, e.g. *must.*
ART	Article	*the, a, an.*
CONJC	Coordinating conjunction	*and, or, nor.*
CONJR	.....	*but* is the only member at the present time.
CONJRO	.....	Those conjunctions which are not members of the other conjunction classes. Temporarily, little use has been made of this class. Forms which belong to it have been assigned to class CONJR2.
CONJR2	Dependency markers	*because, lest, since, unless,* etc. Also such forms as *whereas, whether, while,* which may be reclassified CONJRO.
PREP	General preposition class	This code is used only in the context frame test (Section 2.6). No words in running text receive this tag. If a context frame for a particular preposition class environment is not in the proper table, the system uses the general rule coded PREP.
PREP 0	.....	Those prepositions not included in the other classes.
PREP 1	.....	One member, *to.* The proto-synthex grammar machine programs which operate after the CGC determine if an occurred *to* is part of an infinitive.
PREP OF	.....	One member, *of.*
PN	General pronoun class	This code is used only in the context frame test See analogous remarks about *PREP.*
PN S	Personal nominative pronoun	*I, we, he, they, she,* etc.
PN O	Personal objective pronoun	*me, us, him, them, her,* etc.
PN-S/O	Nominative objective pronoun	*it, you.*
PN-POS	Possessive pronoun	*my, our, his, their, your.*
PN DEM	Demonstrative pronoun	*these, those, this,* etc.

TABLE 1. (*Cont.*)

Label used in Computer Output	*Full Name*	*Examples and Comments*
PN REL	Relative pronoun	*that.*
PN IND	Indefinite reference pronoun	*everyplace, someplace, everybody, noplace, nowhere, anywhere,* etc.
PN RCN	Relative conjunction pronoun	Those relative conjunctions which can function as pronouns in questions. One reason for the creation of this class is that the CGC is used in a question-answering machine, *who, which, what, where, why,* etc.
V/AXV	Verb/auxiliary verb	Those auxiliary verbs which also function as verbs: *can, does, did,* etc. *have* is not included.
•TYPE	Period-type punctuation	. : (Temporary)
,TYPE	Comma-type punctuation	, ; — (Temporary)
/ED/	Forms with *-ed* suffix	*finished, broken,* but not *speed.*
/ING/	Forms with *-ing* suffix	*running, eating,* but not *swing.*
/HAVE/		

MISCELLANEOUS CODING. When possible, nouns, verbs, and pronouns are tagged for number. Nouns ending in 's or s' are tagged as possessives, also. At the present time, the system makes no note of gender.

it. In other cases, such as *beautiful* in *He chooses the beautiful,* the word may be tagged an *adjective.* Suffix Test 2 (Section 2.5) makes this analysis because of the ending *-ful.*

The words in each sentence are put through a battery of independent tests, each of which yields unique or ambiguous code possibilities (see Figure 1). Because the tests are independent, the outputs of each are logically multiplied; the final resulting code is the set of codes all the tests yielded in common. Such results are usually unique. For example, one test might indicate that a word is either a noun or a verb, and another test that it is a verb or an adjective. The only code common to both test outputs is VERB.

In general, when a test provides no information the system assumes that all of the choices NOUN, VERB, ADJECTIVE, are permitted. This prevents the zeroing out of valid data in logical multiplications. It also permits the system to provide some coding for every word. In cases where tests yield incompatible codes, perhaps from an error in a dictionary entry, the CGC prints the tag NONE. Errors indicated by this tag are corrected between runnings of the system.

2.1. *Dictionaries.* Conceptually, the system makes use of a single restricted dictionary of English words and their grammar codes. The actual program uses several smaller English dictionaries (see flow chart, Figure 1). One of these is a function-word dictionary, containing articles, prepositions, pronouns, conjunctions, auxiliary verbs, adverbs not ending in *-ly,* the various forms of the verb *to be,* and the variants of *have.* This dictionary contains under 400 words, and all of its entries have unique grammar codes. There is a separate listing of uniquely coded punctuation marks. The system treats these like function-words.

Finally, there are two separate content word dictionaries containing those

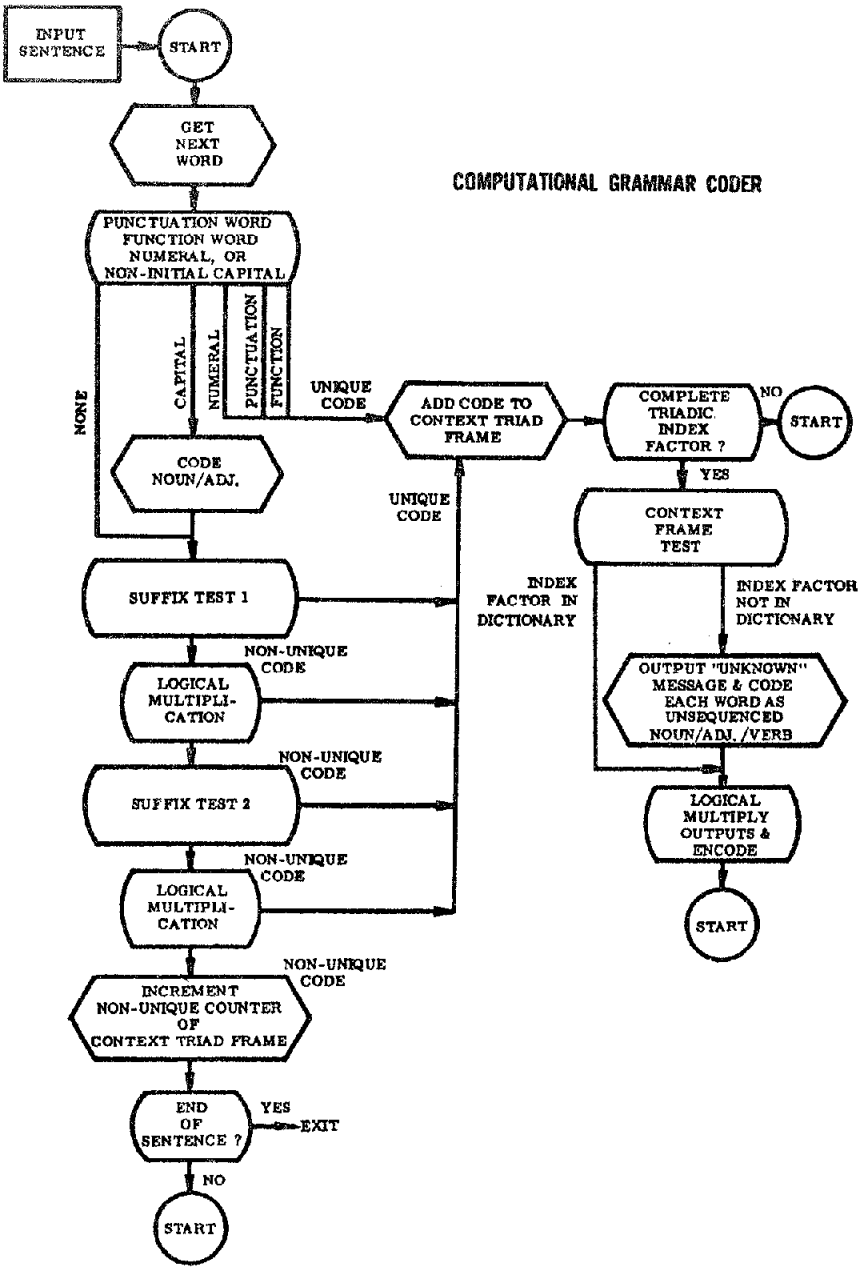

FIG. 1

nouns, verbs and adjectives that are exceptions to the computational rules used in Suffix Test 1 and Suffix Test 2. The total number of words in both of these is under 1500. The codes associated with the content word exceptions are only occasionally unique.

2.2. *Capitalization Test.* The input to the CGC system is essentially unedited keypunched text. Certain modifications are made, however, e.g. special marks indicate paragraphs, and a plus sign is prefixed to words beginning with a capital letter. The Capitalization Test tags non-sentence initial-capitalized words as NOUN/ADJECTIVE ambiguities.

2.3. *Numeral Test.* Sequences of one or more arabic numerals are tagged ADJECTIVE. All numbers could have been treated as a separate class in themselves. Because the CGC can recognize only arabic numerals as numbers, it was decided to code word-classes for them in the same manner as is done for other English words.

2.4. *Suffix Test 1.* The primary function of this test is to extract information from the presence of plural-type endings. With certain exceptions, a word ending in *-s*, *-es*, or *-ies* is either a plural noun or a third person singular verb. Other endings tested for here include *-i*, *-um*, *-is*, *-as*, all indicating noun singular; *-us*, *-ss*, indicating noun singular or third person plural verb or adjective; *-ae*, noun plural; and *-a* indicating a noun of unknown number. Words ending in *-ing* or *-ed* are assigned to classes with the names /ING/ and /ED/. Most exceptions to these rules are in a content-word exception dictionary (e.g. *swing*, *speed*, and *strum*).

Some of the exceptions to the *-es* rule are computed. *-es* occurs as a plural-type ending in preference to *-s* or *-ies* only after written-English representations of spirants. Accordingly, a segmented *-es* is treated as an exception if not immediately presceded by *s-*, *z-*, *h-*, or *x-*.

Forms ending in *-s*, *-es*, and *-ies* are subject to additional special treatment: Suffix Test 1 sends the uninflected form to Suffix Test 2 (see section 2.5), which tests primarily for derivational endings. For example, Suffix Test 1 will recognize *nationalities* as a plural noun and a third person singular verb. It will then strip the word of its *-ies* suffix and add *-y*. Suffix Test 2 will receive *nationality* and code it uniquely as a NOUN because of its *-ity* ending. The system also retains the Suffix Test 1 information that it is a plural form. For other words, such as *babies*, Suffix Test 2 can provide no additional information.

2.5. *Suffix Test 2.* Many of the suffixes utilized in this test are not normally recognizible as such. The sole purpose of this test is to extract whatever grammar code information is present in the last one to five letters of an English word. The last five letters of a word are checked against a list of suffixes, then the last four, three, etc. The following is an extract from the list of suffixes ending in the letter *-l*. The list is alphabetized in reverse order.

l-, NOUN/VERB	*ladna-*, NOUN
la-, ADJECTIVE	*le-*, NOUN/VERB
laa-, NOUN	*li-*, NOUN/VERB
labm-, NOUN	*luf-*, ADJECTIVE

The few words whose grammar codes are exceptions to those rules are listed in a content word exception dictionary.

2.6. *Context Frame Test.* This test is different from the others in that it

operates on units larger than a single word. The test is called for whenever the CGC system has discovered a string consisting of two uniquely coded words bracketing one or more ambiguously coded forms. Such a string permits the use of a triadic index factor composed of:

(1) the numerical representation of the grammar code of the left uniquely coded word,
(2) the number of non-uniquely coded words,
(3) the numerical representation of the grammar code of the right uniquely coded word.

For example:

ARTICLE	ADJECTIVE VERB	NOUN ADJECTIVE	VERB
(uniquely coded)	(non-uniquely coded)		(uniquely coded)
3.	2.		4.

This value is used in a binary search of a context triad frame table containing information about the permissible sequences of codes that may fit between the unique codes. Thus the table entry for "3.2.4" contains the three sequences that may fit between article and verb:

ADJECTIVE – NOUN NOUN – ADVERB NOUN – NOUN

Enough information exists in this example to determine a unique code for each word. The computation is complex. The logical multiplication of test values is first carried out for one word at a time, as between the outputs of other tests. However, when a context frame test code is found in a disjunction (not common to the outputs of both tests), its sequential partner is also eliminated as a choice.

In this example, previous tests may have resulted in:

ADJECTIVE	NOUN
VERB	ADJECTIVE

with four possible sequences implied.

The context frame test yielded three sequences:

ADJECTIVE	–	NOUN
NOUN	–	ADVERB
NOUN	–	NOUN

Logical multiplication of the sets of left-hand codes eliminates both NOUN choices in the results of the context frame test:

ADJECTIVE	–	NOUN
....	–	ADVERB
....	–	NOUN

But since these were members of *sequences*, the elimination of the corresponding right-hand codes is implied. The only remaining sequence is then ADJECTIVE-NOUN.

The context triad frame table contains approximately 500 entries. The maximum number of ambiguously coded words in a sequence that it can handle is three. This means that if the middle component of the index is greater than three, no information will be contained in the table.

The table entries do not handle all possible cases, even within the bounds of

this limitation. The maximum possible number of cases (including ones which never occur in written English) would be the number of grammar codes recognized, multiplied by the maximum number of ambiguously coded words in a sequence the system can handle, multiplied by the number of grammar codes recognized, i.e. $30 \times 3 \times 30 = 2700$. Nevertheless, only about 500 appear with any great frequency in English text.

The entries actually in the table were empirically derived by hand analysis of a sample of Golden Book Encyclopedia text. When the number of entries appeared to account for approximately 90 per cent of the encountered text, it was decided to automate the process of additional entry derivation.

The CGC system now prints triadic index factors called for by analysis of text but not found in the table. Missing entries may be added after each run of the program. In the particular experiments run, the context frame test was used at least once per sentence. The need for this test is diminished in text whose vocabulary is rich in words with derivational suffixes.

3. *Formal Description of the System*

The preceding discussion has been devoted to the description of the operators or tests used by the CGC system. The flow chart in Figure 1 shows their interrelations in the operating program. The more formal description presented in this section[3] shows that the basis for computing grammar codes for the words in a sentence is one of successively reducing the number of combinatorial choices of word-class codes. At early levels in the system the operators reduce the choice from thirty codes per word to four or fewer (see sections 2.1 through 2.5). At the level of the triad frame test, the system operates to reduce the number of permissible combinations of codes for strings of words bounded by words with single codes. The bounding always occurs since the beginning and ending of a sentence must always be uniquely coded in the system. The net result of all tests is to minimize the number of codes applied to each word in the sentence.

For the sake of simplicity, the following formalization applies to an idealization of the CGC which will be called *coder*. The coder operates on units not larger than a sentence. It assumes that the system of grammar tables and dictionaries are complete. (Although this assumption is not strictly true for the operating program, the system feeds back error messages which lead eventually toward achieving the truth of the condition.)

Definition. A *coder system* is a 5-tuple

$$S = (\Sigma, \{w_i\} 1 \leqq i \leqq r, D, \{g_i\} 1 \leqq i \leqq 5, T$$

which has the following properties:

(i) Σ is a finite, nonempty set (the basic alphabet)

(ii) $\{w_i\}$ $1 \leqq i \leqq r$ is a finite sequence of Σ-words[4]

[3] We are indebted to Seymour Ginsburg, of the System Development Corporation, for the formalization presented here.

[4] Given a finite nonempty set Σ, a Σ-word is any finite string or sequence of given symbols from Σ.

(iii) D is a finite, nonempty set (of "grammar codes")

(iv) Each g_i is a mapping of $\{w_j / 1 \leqq j \leqq r\}$ into $N(D)$, $N(D)$ being the family of all nonempty subsets of D such that for each Σ-word w_i

$$g(w_i) = g_1(w_i) \cap g_2(w_i) \cap g_3(w_i) \cap g_4(w_i) \cap g_5(w_i)$$

is nonempty

(v) $\#[g(w_1)] = \#[g(w_r)] = 1$[5]

(vi) T is a function which yields for every triple (A, B, k)—A and B being non empty subsets of D, k being a positive integer—a set of k-tuples $(x_1, \cdots, x_k)$, each x_m in D.

For each coder system S we now define a function γ_S from a certain subset of he first r integers to the family of all subsets of D.

(α) Let i be an integer for which $\#[g(w_i)] = 1$. Define $\gamma_S(i)$ to be $g(w_i)$.

(β) Suppose that i is an integer for which $\#[g(w_i)] > 1$ and $\#[g(w_{i-1})] = 1$. Let $j+1$ be the smallest integer greater than i such that $\#[g(w_{j+1})] = 1$. The integer j exists since $\#[g(w_r)] = 1$. Let

$$F(i) = T(g(w_{i-1}), g(w_{j+1}), j - i + 1) \cap [g(w_i) \times g(w_{i+1}) \times \cdots \times g(w_j)].$$[6]

Define $\gamma_S(i)$ to be the set $\{x_i / \text{there exists some tuple } (x_i, \cdots, x_{j-i+1}) \text{ in } F(i)\}$.

(γ) $\gamma_S(i)$ is undefined for those i occurring in neither (α) nor (β).

In the preceding description:

1. g_i refers to the ith test as described in sections 2.1 through 2.5.
2. T represents the Context Frame Test as described in section 2.6.
3. $g(w_i)$ represents the results of the intersection of the outputs of the tests described in section 2.1 through 2.5., i.e.

$$g(w_i) = g_1(w_i) \cap g_2(w_i) \cap g_3(w_i) \cap g_4(w_i) \cap g_5(w_i).$$

4. Condition (v) in the preceding, $\#[g(w_1)] = \#[g(w_r)] = 1$, refers to the fact that the first and last elements in a sentence always have only one grammar code; these elements are the markers of the beginning and end of the sentence.
5. $T(g(w_{i-1}), g(w_{j+1}), j-i+1)$ represents the $j-i+1$-tuples of permissible grammar codes as derived from the Context Triad Frame; where $g(w_{j+1})$ is the right unique code, and $j-i+1$ is the number of non-uniquely coded words in the middle.
6. $g(w_i) \times g(w_{i+1}) \times \cdots \times g(w_j)$ represents the cartesian product of the grammar codes obtained for words 1 through j just before the application of the Context Frame Test; i.e. the $j-i+1$-tuples of permissible grammar codes as described in sections 2.1 through 2.5.
7. $\gamma_S(i)$ represents the grammar codes of the ith word in a sentence after the application of the tests described in sections 2.1 through 2.6.

[5] Given a set A, by $\#(A)$ is meant the number of elements in A.

[6] Given sets $A_1, \cdots, A_m$, by $A_1 \times \cdots \times A_m$ is meant the set $\{x_1, \cdots x_m) / x_i \text{ in } A_i \text{ for each } i\}$.

4.0 *Empirical Testing of the System*

As described above, the CGC system was developed empirically by the hand analysis of the simple text found in a child's encyclopedia. When it was run on several pages from that encyclopedia, it correctly and unambiguously tagged slightly over 90 per cent of the words. Forty-five per cent of all words were uniquely tagged by the function word dictionary. The remaining 45 per cent owe their unique tags to the application of more than one test. Almost all of such words received unique tags as a result of the intersection of ambiguous codes which were outputs of suffix tests and the context frame test. Of the remainder, 3 to 4 per cent were outright errors. Almost all of the ambiguous tagging and errors were due to mistakes that had been made in dictionary entries, context triad frames, or the lack of appropriate context triad frames for particular situations. A few of the ambiguities are removed by higher level syntactic analysis accomplished by other parts of the parsing system. However, a certain amount of ambiguity is inherent in any analysis of English that ignores meaning, and the presence of such ambiguity forces the analysis of multiple tree structures.

Analysis of the operation of the system indicated that in some senses the scientific text was actually easier to analyze than the child's encyclopedia.

TABLE 2. ANALYSIS OF *Scientific American* TEXT

Word	*Class*	*Number*	*Word*	*Class*	*Number*
Considering	/ING/		at	PREP O	
that	PN REL		first	ADJ	
returning	/ING/		glance	NOUN/VERB	SING/PLU
space	NOUN	SING	.	. TYPE	
vehicles	NOUN	PLU	Meteors	NOUN	PLU
will	AXV		are	VERB IS	PLU
be	VERB IS		slowed	/ED/	
entering	/ING/		by	PREP O	
the	ART		the	ART	
air	NOUN	SING	atmosphere	NOUN/ADJ	SING
almost	ADV		,	, TYPE	
as	CONJR2		but	CONJR	
fast	ADJ		at	PREP O	
as	CONJR2		rates	NOUN	PLU
meteors	NOUN	PLU	far	ADJ	
do	V/AUX		beyond	PREP	
,	, TYPE		the	ART	
the	ART		tolerance	NOUN	SING
feasibility	NOUN	SING	of	PREP OF	
of	PREP OF		any	ADJ	
atmospheric	ADJ		human	ADJ	
deceleration	NOUN	SING	occupant	ADJ/NOUN	/PLU
is	VERB IS	SING	of	PREP OF	
by	PREP O		a	ART	
no	ADJ		space	NOUN	SING
means	VERB	SING	vehicle	NOUN	SING
obvious	ADJ		.	. TYPE	

Although scientific sentences were longer and more complex, the high frequency of auxiliary verbs and of words with recognizible suffixes actually tended to improve the operation of the CGC. Original fears that sequences of four or more unidentified parts of speech would occur with great frequency were not substantiated in fact.

Table 2 shows the results for tagging a small segment of the *Scientific American* text. Table shows the same for a segment for the *Encyclopedia Americana*. The sentences chosen are fairly typical in that they include the most frequent errors that the system makes. They are neither particularly easy nor difficult sentences in terms of the CGC operation.

In the second column, third word of Table 2, the system tagged *glance* as either a noun or a verb. This ambiguity can only be removed at a higher level in the grammar machine after the system recognizes the dependent phrase. Six words from the bottom of the second column in Table 2, another type of ambiguity occurs. In this case, the word *occupant* in the phrase *of any human occupant*, is tagged ambiguously as an *adjective* or a *noun*. This is an unnecessary ambiguity, since the frame *adjective* ..... *prep. of* can contain only a noun or a verb. Consequently a change in the context triad frame dictionary will eliminate this type of inadequacy in the system. (The output of Suffix Test 2 precluded the verb choice in this example.) A change in the context triad frame dictionary will also correct the mistake made in the coding of *means* in column 1 of Table 2.

Similarly, in the examples shown in Table 3, some of the ambiguities and errors can be removed by minor changes in dictionary entries. But as indicated earlier, some are truly ambiguous grammatical constructions whose ambiguity can only be resolved at higher levels of syntactic and even semantic analysis. At the present stage of development, the system contains many unnecessary ambiguities, but continued running on large samples of text will bring these to light and result in corrections and improvements. At the early stage in a rapidly improving developmental system a complete statistical analysis of errors and ambiguities has not been considered worthwhile. However, as the system reaches a plateau where improvements are not so obvious, such detailed analyses will be made and the output of the CGC will be compared with the output of a dictionary lookup for each word in the text. This comparison will show to what extent the tagging of words in context eliminates form-class ambiguities as tagged in the dictionary.

Even from these early findings in running the CGC system on scientific text, we have observed that it is accurate enough to form a satisfactory first module in our syntactic analysis system. The syntactic analyzer of which it is a subsystem has been programmed and checked out. This higher level syntactic analysis system has been constructed to work with the types of codes and ambiguities natural to the CGC. In addition, at the phrase and clause level of analysis, several routines are available to resolve certain types of noun-verb or noun-adjective ambiguity.

We are aware, particularly from the work done by Oettinger [6], that there is a great deal of ambiguity inherent in the grammatical structure of English sentences. If only structural cues are used, much of this ambiguity probably cannot

TABLE 3. ANALYSIS OF *Encyclopedia Americana* TEXT

Word	*Class*	*Number*
The	ART	
blue	ADJ	PLU
Whale	ADJ/VERB/ NOUN	SING
has	/HAVE/	SING
a	ART	
massive	ADJ/NOUN	/SING
head	NOUN/VERB	SING/PLU
and	CONJC	
broad	ADJ	
snout	NOUN	SING
,	, TYPE	
and	CONJC	
the	ART	
body	ADJ/NOUN	/SING
tapers	NOUN/VERB	PLU/SING
gradually	ADV	
to	PREP 1	
the	ART	
flukes	NOUN	PLU
.	. TYPE	
The	ART	
dorsal	ADJ	
fin	NOUN	SING
is	VERB IS	SING
falcate	NOUN	SING
and	CONJC	
less	ADJ	
than	CONJR2	
a	ART	
foot	NOUN	SING
long	ADJ	
,	, TYPE	
and	CONJC	
is	VERB IS	SING
situated	/ED/	
at	PREP O	
a	ART	
point	NOUN	SING
a	ART	
little	ADJ	
more	ADJ	
than	CONJR2	
three-fourths	NOUN	PLU
the	ART	
distance	NOUN	SING
from	PREP O	
the	ART	
top	NOUN	SING
of	PREP OF	
the	ART	
snout	NOUN	SING
to	PREP 1	
the	ART	
notch	NOUN	SING
of	PREP OF	
the	ART	
flukes	NOUN	PLU
.	. TYPE	

be removed. The eventual solution for practical language systems may have to wait on the development of techniques for using semantic cues for eliminating structural ambiguity. How serious the problem of ambiguity will be for any particular application of language processing still remains to be discovered. But it is already apparent that syntactic ambiguity poses one of the most challenging of research problems in computer analysis of natural language.

REFERENCES

1. ANDREWS, R. C. Whale to whaling. *Encyclopedia Americana 29* (1961), 231–234.
2. BECKER, J. V. Re-entry from space. *Sci. Amer. 204*, (Jan. 1961), 52–57.
3. HARRIS, Z. S. Computer syntactic analysis. Transformations and Discourse Analysis Papers No. 15. Reports to the National Science Foundation, U. of Pennsylvania, Phila.
4. LAMB, S. M. Segmentation. Proceedings of the National Symposium on Machine Translation, 335–342. Prentice-Hall, Englewood Cliffs, N. J., 1961.
5. LINDSAY, R. K. The reading machine problem. Unpublished doctoral dissertation, Carnegie Inst. Tech., 1960.

6. OETTINGER, A. G. Automatic language translation. Harvard University Press, Cambridge, Mass., 1960.
7. SALTON, G.; THORPE, R. W. An approach to the segmentation problem in speech analysis and language translation. Paper, First Internat. Conf. Machine Translation of Languages and Applied Language Analysis, Teddington, England, Sept. 1961.
8. SIMMONS, R. F., KLEIN, S., AND MCCONLOGUE, KEREN L. Toward the synthesis of human language behavior. *Behav. Sci.* (July 1962), 402–407.
9. YNGVE, V. H. The feasibility of machine searching of English texts. Proc. Internat. Conf. Scientific Information, Washington, D. C., 1959, 975–995. Nat. Acad. Sciences-Nat. Res. Council.