REPRINTED FROM
ADVANCES IN COMPUTERS, VOL. 8

ACADEMIC PRESS INC., NEW YORK

Syntactic Analysis of Natural Language

NAOMI SAGER

Institute for Computer Research
in the Humanities
New York University
New York

1. Linguistic Basis for Computations

Writers of programs for handling natural language material, whether for mechanical translation, automatic indexing, or other purposes, have learned (sometimes the hard way) that without grammatical analysis their efforts at processing language data reach an early limit. Thus, persons whose original goals may have been far afield find themselves concerned with automatic syntactic analysis, and, as they become further acquainted with the problem, with different grammatical theories, since no piecemeal assemblage of grammatical rules will suffice for treating the coherent and productive, yet complex and detailed, mechanism of language.

The main styles of grammatical description can by now each be associated with an existing program or program-in-the-making.

However, not all discussion of the computability of language has been related to program development; for example, papers [*2*, *37*] treat the application of the notion of grammatical categories—familiar from the Polish logicians—to language word classes, paralleling what linguists had already been doing, i.e., defining word classes on syntactic grounds. The notion of grammatical categories was related by Hiż to an early form of linguistic string analysis [*26*] and to transformational analysis [*24*].

1.1 Immediate Constituent Analysis

Several discussions and programs are based on immediate constituent analysis (ICA), or developments therefrom [*21–23*, *27*, *43*, *44*]. ICA is similar to the method of parsing and traditional methods of syntactic analysis, which segment a sentence into parts (e.g., noun phrase and verb phrase) and these in turn into smaller parts, and so on. It was formulated by Leonard Bloomfield and others [*3*, *17*, *51*], and has been formalized under the name phrase-structure analysis by Chomsky [*9*]. A formal theory of grouping as a basis for ICA is given by Hiż [*25*].

The weak equivalence of several other types of grammars used by syntactic programs (e.g., dependency grammar of Hays and projective grammar of Lecerf [*38*]) to an immediate constituent grammar is shown by Gross [*13*]. Despite progress in this line of program development, there is not as yet an efficient IC analyzer which covers the bulk of English grammar, provides a reasonably small number of analyses per sentence and is not limited as to the length of sentences it analyzes. This is partly due to certain inherent difficulties of constituent analysis for computational purposes.

1.2 String Analysis

These deficiencies do not exist in linguistic string analysis [*18*]. The main relevance of string analysis for computational purposes is that it overcomes the problem of discontinuity in a natural way and provides a framework for introducing further linguistic refinements without adding appreciably to the bulk or complexity of the grammar. These features are both due to the fact that the linguistic string is the least segment of a sentence with respect to which grammatical restrictions can be stated. For example, a current string analysis program [*45*], using several hundred rules (in this case, strings and restrictions) would require the addition of several hundred restrictions framed on the existing strings, in order to refine it by the main transformational

subclasses. If we consider a program without a string framework, the leading program, the Harvard Predictive Analyzer [*34, 35, 40*] using currently several thousand rules, is estimated by Kuno [*33*] to require an order of magnitude increase in the number of grammar rules in order to achieve a considerable refinement.

A program using an early form of string analysis ran successfully on the Univac I at the University of Pennsylvania in 1959 [*20*]. The Univac program incorporated the main constructions of English grammar, not all in equal detail, and obtained one preferred analysis of a sentence. It was utilized for the syntactic part of one of the earliest question-answering programs (Baseball) [*12*]. Using string analysis, it was shown [*46*] that the fully nested structure of English made possible the utilization of a push-down store technique in a left-to-right word-by-word analysis of a sentence, to produce all syntactic analysis of the sentence in a single scan. The left-to-right procedure for string analysis, as programmed for the 7094, obtains all analyses of a typical sentence (of a scientific text) in about 5 seconds. These number 1–5 per sentence, given the convention (which can be lifted) that permanent predictable ambiguities which can be read off existing outputs are not printed. Most often, the first analysis is the one that expresses the author's intended meaning. Another string program [*7*], which uses a canceling and cycling automaton form of string grammar [*15*], is in use for analyzing narrative medical text.

1.3 Left-to-Right Scanning and Predictive Analysis

The left-to-right approach was used in an essential way in predictive analysis, developed by Ida Rhodes in the National Bureau of Standards program for syntactic analysis of Russian [*1, 41, 42*]. Since 1961, a push-down technique has been combined with predictive analysis in the Harvard program for syntactic analysis of English (HPA). The HPA goes beyond the limitations of IC programs by using a set of rules each of which specifies a structure up to the end of that structure (from the point at which the rule applies) and places each structure in respect to the analysis of the sentence up to the point at which the rule is used. This enables the HPA to insert material nested inside of a structure while keeping track of the still expected residue of the structure.

During this time, in the field of compiler writing, the single-scan left-to-right approach to scanning an input string of a formal language, and other syntax-oriented approaches, began appearing in the literature [*6, 28, 29*]. Recent developments suggest that syntax-oriented compilers may be developed to accept languages having essential natural language mechanisms.

1.4 Transformational Analysis

There remains the linguistic method of transformations [*8*, *9*, *16*, *19*] which, while not easily computable, provides a much subtler analysis and brings together on syntactic grounds semantically equivalent or related sentences. Joshi [*30–32*] has designed an algorithm (machine independent) for transformational decomposition, and is writing a substantial portion of a transformational grammar of English in a form suitable for the algorithm. His procedure is entirely based on some rather basic and quite general properties of transformations and does not depend on any prior analysis, such as immediate constituent, or string analysis, above. Several programs in the specifically generative style of transformations [*10*] have been constructed. Matthew's method [*39*] was to analyze by synthesizing sentence strings from transformational rules, though the program did not reach a practical working stage. Walker and associates [*49*, *50*] are constructing a transformational program as part of an information retrieval system. As yet this program covers only certain portions of English, but it is to be hoped that the system will be extended. There exist related studies around the problem of the computability of generative structures and transformations. Among these we might mention Yngve's treatment of discontinuous elements [*14*, *53*].

1.5 Other Methods

In the work on natural language, there are still only a few programs which attempt to incorporate a substantial portion of the grammar of the language. A number of existing programs (not mentioned here) make one or another simplification of the problem. As an example of a program which includes a large-scale computer grammar of a language quite different from English we note the work at Berkeley on Chinese [*11*], which is related to methods of Lamb [*36*]. Reference should also be made to COMIT, a programming language designed for syntactic treatment of natural language [*52*]. Several survey articles have appeared: Simmons [*48*], on question-answer programs, though some idea of the full linguistic complexity of questions may be obtained from Bolinger [*5*]; Bobrow [*4*], a general survey of computer programs for syntactic analysis of English.

The remaining two sections of this paper describe in greater detail the left-to-right procedure for string analysis of sentences (Section 2) and the computer program based on it which is now in operation (Section 3). The description emphasizes the means of handling such essential

features of language structure as detailed subclass restrictions, coordinate and comparative conjunctions, and syntactic ambiguity. The program was developed as part of the National Science Foundation Transformations and Discourse Analysis Project at the University of Pennsylvania.

2. A Procedure for Left-to-Right String Decomposition of Sentences[1]

2.1 Word-by-Word Formulation of String Structure

The procedure described in this section is based on an axiomatic formulation of linguistic string theory which presents, in terms of particular syntactic categories for words of the language (e.g. N noun, tV tensed verb), a set of elementary strings of word categories and rules for combining the elementary strings to form sentences. An example of an elementary string is $N\ tV$ (*Power corrupts*). Recognizing the structure of a sentence will mean decomposing the sentence into elementary strings of the language, each elementary string being defined as a sentence, or as entering at a stated point of another elementary string in a sentence.[2] The procedure calls upon the strings and restrictions (described below) of a particular language but is itself independent of the particular grammatical material used. One is able to change details of the grammar without altering the program structure. One might even expect that such a program would be usable for other languages by changing the entire grammatical content, i.e., the list of strings and restrictions.

The strings are groupable into classes based on how and where they can be inserted into other strings. If $\xi = X_1 \cdots X_n$ is a string, X ranging over category symbols, the following classes of strings are defined:

l_X Left adjuncts of X: adjoined to a string ξ to the left of X in ξ, or to the left of an l_X adjoined to ξ in this manner.

r_X Right adjuncts of X: adjoined to a string ξ to the right of X in ξ, or to the right of an r_X adjoined to ξ in this manner.

n_X Replacement strings of X: adjoined to a string ξ replacing X in ξ.

s_ξ Sentence adjuncts of the string ξ, adjoined to ξ at any interelement point or to the left of X_1 or to the right of X_n, or

[1]Talk presented at the National Science Foundation Seminar on Documentation Research, Washington, D.C., March 6, 1962.

[2]We use the term "elementary string in a sentence" to designate the sequence of words in a sentence which correspond to an elementary string of word categories (string of the grammar). Henceforth "string" is to mean "elementary string" unless otherwise indicated.

to the right of an s_ξ which has been adjoined to ξ in one of these manners.

c_ξ Conjunctional strings of ξ, conjoined to the right of X_i in ξ $(1 \leq i \leq n)$ or to the right of a c_ξ conjoined in this manner.

z Center strings, not adjoined to any string.

There are various restrictions on the repetition and the order of various members of the classes of adjuncts.

Roughly speaking, a center string is the skeleton of a sentence and the adjuncts are modifiers. An example of a left adjunct of N is the adjective *green* in *the green blackboard*. A right adjunct of N is the clause *whom we met* in *the man whom we met*. A replacement string of N is, for instance, *what he said* in the sentence *What he said was interesting*. The same sentence with a noun instead of a noun-replacement string might be *The lecture was interesting*. Examples of sentence adjuncts are: *in general, at this time, since he left*. The C-strings, which will be described below, have coordinating conjunctions at their head. An example is *but left* in *He was here but left*. Examples of center strings are *He understood* and also *We wondered whether he understood*.[3]

The words of the language are assigned to one or more word categories on the basis of their grammatical properties in the language as a whole; e.g., *the* has the one assignment T (article); *move* has three alternative assignments, N (noun) $/tV$ (tensed verb) $/V$ (untensed verb). Therefore, to the sequence of the words of a sentence there corresponds a family of one or more sequences of categories, one per word. We call each such sequence of categories a representation of the given sentence. The theory now asserts that every sentence in the language has at least one representation which is identical to a string in class z, or to one upon which there have been carried out adjunctions, replacements, or conjunctions, as provided in the above definitions.

In the terms of the string-class definitions above, we can now define an analysis of a sentence. We will say that a sentence representation is analyzed if we can assign every symbol in that sentence representation to some elementary string, either a center string, or an adjunct or replacement string inserted according to the operation of the string-class definitions. Each analyzed sentence representation is a different

[3]In this formulation of string grammar, the occurrence in a sentence of a subject or object which does not consist of a word category with its adjuncts is treated as the result of N-replacement; e.g., *I know that he was there* from *I know N* (*I know something*). This treatment is problematical for the few verbs which do not occur with an N object, e.g., *wonder*: *I wonder whether he was there*, $\nexists$ *I wonder something*. This difficulty does not arise if the elementary strings are allowed to have strings as elements, e.g., Σ for subject strings, Ω for object strings, yielding an assertion center string $\Sigma tV\Omega$ (t = tense). This is the approach adopted in the machine grammar.

syntactic analysis of the sentence whose successive words belong, respectively, to the successive categories of the analyzed sentence representation.

To set the stage for recognizing the structure of a sentence, we observe the following, as a direct consequence of the above: For a given sentence, there exists at least one sentence representation such that, if we proceed from left to right through a sentence, each successive word belongs to a category which is either the continuation of an unfinished string, or is the beginning of a new string permitted at that point in the sentence representation by the string-class definitions.

We start out by requiring the occurrence of a center string, since the theory asserts that every sentence has a center string. To illustrate, suppose that the grammar contains only a single center string, $N\ tV\ N$, and a single adjunction class r_N containing a single string $P\ N$.

Grammar:

$$N\ tV\ N \in z$$

$$P\ N \in r_N$$

Suppose the sentence to be analyzed is

Cars without brakes cause accidents.

$$N \quad P \quad N/tV\ N/tV/V \quad N$$

The first symbol N of the required center string $N\ tV\ N$ matches the first sentence symbol N. The sentence word *cars* is thereby analyzed as the first element of the center string. [In a richer grammar the symbol N might be the beginning of some other string permitted in the sentence-initial position; in that case, there would be a second analysis at this point.] The second sentence word does not have a classification which matches the current (second) symbol of the center string. However, it does have a classification P which is the first symbol of a string $P\ N$ permitted at this point as a right adjunct of N. Accepting this analysis of the second sentence word, the program inserts in the list of requirements the string $P\ N$ (with P already satisfied) and pushes down the remaining requirements of the center string, to return to them when the $P\ N$ string is finished. We proceed in the same way for the $P\ N$ string as for its host, the $N\ tV\ N$ string; i.e., we ask at each word position: Does one of the category assignments of the current sentence word match the current symbol of the string in progress? If it does, we advance in that string. If it does not, we ask: Is it the beginning of some string which is permitted at this point? If so, we begin a similar process at the next level by inserting the remaining symbols of the string into the list of requirements already established.

The procedure produces an analysis of a sentence if, by accepting one match at each position, it reaches the end of the sentence with no requirement unsatisfied. Since n matches at the ith position means that we have n potential analyses of the sentence which are identical up to the ith position but different from i and on, we can obtain all analyses of the sentence either by carrying different analyses in parallel or by stacking alternative analyses at each branch point and following them through in some prescribed order. In the example sentence, the string $P\,N$ (*without brakes*) is the only interruption in the $N\,tV\,N$ center string (*cars cause accidents*), and there is only one analysis.

2.2 Indirectly Defined Strings: *wh* and *C*[4]

Two classes of strings require additional description. The first of these, the class of *wh* strings (relative clauses), has members which are identical (except for the prefixed *wh* word) with a large number of already defined strings, in all but one (or two) symbol position(s) of the string. It is therefore convenient to obtain these strings in the process of computation by an operation (called "omission") on strings already defined. The operation of omission consists of skipping one required N (or $P\,N$) from a string ξ or from certain adjuncts of ξ. For example, from the center string $N_1\,tV\,N_2$ (*The man wrote the book*) two *wh* strings can be obtained by omitting, respectively, N_1 and N_2: *wh* $tV\,N_2$ (*who wrote the book*) and *wh* $N_1\,tV$ (*which the man wrote*). When the *wh* string adjoins a host N, the omitted N can be identified with the host N; e.g. the *wh* string from which N_1 was omitted adjoins N_1 (*the man who wrote the book*). The omission operation may also be applied to N in certain adjunct strings, mainly $P\,N$. For example, from $N_1\,tV\,N_2\,P\,N_3$ (*The author chose the title of the book*) in which $N_1\,tV\,N_2$ (*The author chose the title*) is the center string and $P\,N_3$ (*of the book*) is an adjunct of N_2 (*title*), we can form three *wh* strings which can adjoin N_3: (1) *wh* $N_1\,tV\,N_2\,P$ ((*the book*) *which the author chose the title of*), omitting N_3; (2) P *wh* $N_1\,tV\,N_2$ ((*the book*) *of which the author chose the title*), omitting $P\,N_3$; and (3) $N_2\,P$ *wh* $N_1\,tV$ ((*the book,*) *the title of which the author chose*), omitting N_2 with its adjunct $P\,N_3$.

Another class of strings which is best defined by an operation is the class of C strings headed by coordinate conjuctions *and, or, but,* etc. If listed, this class would contain all of the strings of the grammar, plus virtually all parts of all strings (each preceded by C), which makes a very large class. But at any point in a sentence the choice of C string

[4]To justify going into some linguistic detail here, we note that the procedure would not function without a considerable organization of the linguistic data.

must be made from a limited subset of C strings, depending on which strings precede C in the analyzed portion of the sentence.

One way to define these C strings is as follows: if α and β are strings in class x, then $\alpha\, C\, \beta$ is a C equivalence and the portion $C\, \beta$ is a C string. For example, let x be the class of center strings, with α the assertion center and β the question. $\alpha\, C\, \beta$ gives us such sentences as *It is a fact and who can deny it.* Or let $x =$ the right adjuncts of N; an example is *the delegates elected but not receiving the requisite majority of the votes.* When $\alpha = \beta$ (that is, when the same string ξ appears on both sides of the C equivalence), shortened forms $\xi_1\, C\, \xi_2'$ and $\xi_1'\, C\, \xi_2$ may occur, where ξ' is structurally identical to ξ except that certain elements present in ξ are absent in ξ'; but the first element of ξ_1 is never absent. For example, *I came and I went* has a shortened form *I came and went*, and *to formulate the question and to answer the question* has a shortened form *to formulate and to answer the question.* The elements which are absent in ξ' can be shown to be zero occurrences of the words which satisfy the corresponding elements of ξ. Shortened forms $\xi_1'\, C\, \xi_2'$ can also be obtained by the above rule: *to formulate and answer the question.* An operation similar to omission can be defined in order to obtain C strings with zeroed elements; and, if desired, the zeroed words can be filled in: *I came and (I) went.*

Alternatively, the shortened forms can be defined without reference to the zeroed elements. The most frequent case is one in which the zeroed elements in a C equivalence $\xi_1\, C\, \xi_2$ form an unbroken sequence around C from a point a in ξ_1 to a point b (determined by a) in ξ_2. This includes sequences with zeroing only in ξ_1 or only in ξ_2 or with no zeroing at all. The C strings in this case can be defined as follows: Let $\xi = X_1 \cdots X_n$. If a conjunction C appears following X_l in an occurrence of ξ in a sentence ($1 \leq l \leq n$), then the string headed by C which can interrupt ξ at this point is any one of the following:

$$CX_l; \quad CX_{l-1}X_l; \quad CX_{l-2}X_{l-1}X_l; \quad \cdots; \quad CX_1 \cdots X_l$$

For example, let $\xi =$ *to* $V\, \Omega$ (*to repeat the question*). In a sentence which begins *To repeat the question and answer* $\cdots$, $l = 3$, and corresponding to $C\, X_l$ we have *to* $V\, \Omega\, C\, \Omega$ in which *question* and *answer* are both seen as N. Another analysis (corresponding to $C\, X_{l-1}\, X_l$) of the word sequence is *to* $V_1\, \Omega_1\, C\, V_2\, \Omega_2$ in which *answer* is taken as a verb, with $\Omega_2 = 0$. For the same ξ, an example of $C\, X_l$ where $l = 2$ would be *to* $V\, C\, V\, \Omega$ (*to formulate and answer the question*).

There are also cases in which the zeroed elements are final or interior portions of ξ_2. The C strings in these cases always appear after the complete ξ_1 and can be characterized in respect to the structure of ξ_1.

The main C strings of this type are listed here for $\xi_1 = \Sigma\ t\ V\ \Omega$, in which Σ = subject, V = untensed verb, t = a tense suffix or auxiliary (including forms of *have*, *be*, *do*), and Ω = object.

Σ	t	V	Ω	C Σ	*He laughed and she also.*
Σ	t	V	Ω	C Σ t	*He laughed but she didn't.*
Σ	t	V	Ω	C Σ Ω	*He chose a rose and she a poppy.*
Σ	t	V	Ω	C t	*He should have gone but didn't.* also *but hasn't, but couldn't.*
Σ	t	V	Ω	C $\overline{D}$	*He left and fast.* ($\overline{D}$ = a subset of verb and sentence adjuncts)

2.3 Restrictions

A string is a sequence of word categories, but not every combination of words drawn from the respective categories makes a satisfactory string occurrence in a sentence. Sometimes only words having related grammatical properties are acceptable in the same string, or in adjoined strings. We define subcategories to express these grammatical properties. Then particular strings will carry restrictions as to the subcategories of words which can occur together as elements in the same string or of related strings. For example, the fact that a plural noun cannot occur as the subject of a singular verb constitutes a restriction on the $N\ tV\ N$ center string. Restrictions could be entirely eliminated if we were willing to increase the number of categories and strings by a very large amount. For example, instead of the restriction on subject-verb agreement as to number, we could list every string to which this restriction applies, writing the string once in its singular and once in its plural form. However, in so doing we would not only double the number of these strings and increase the number of categories but we would mask the relationship between the strings of each pair.

The machinery for applying restrictions in a recognition program is based on the observation that restrictions operate within a particular scope: (1) either within some string ξ, or (2) between some element of a string ξ and the head of φ where φ is an adjunct of ξ or a replacement string inserted into ξ, or (3) between the heads of two φ adjoined to the same ξ and related to each other in one of a few specified ways. Thus in the recognition procedure it becomes a relatively simple matter to locate the arguments of a restriction. One moves from a particular symbol to one of the members of its adjunct classes (or vice versa) or to one of the other elements of the given string, or among related adjuncts of the given string.

2.4 Syntactic Ambiguity

Syntactic ambiguities stem from several sources: from multiply classified words, like *increase* which can be a noun or a verb, and also from the possibility of the same symbol sequence resulting from different applications of the string-class definitions. For example, in the sentence *People wearing hats is unusual*, the segment *people wearing hats* can be assigned to only one string, a noun-replacement string agreeing with the singular verb *is*. In the sentence *People wearing hats are unusual* the same segment is assigned to different strings, as a plural noun (*people*) with a right adjunct (*wearing hats*) where the plural noun agrees with the plural verb *are*. The sentence *People wearing hats can be unusual* is ambiguous. Since the verb *can be* does not distinguish number, both above assignments of the segment *people wearing hats* are grammatical. [One might think that *people wearing hats* can also be taken as a compound noun, like *feather bearing hats*. This requires accepting *hats wear people* as grammatical.]

We distinguish temporary ambiguities which arise in the course of computation but are resolved when the entire sentence is analyzed, from permanent ambiguities, i.e., more than one syntactic analysis of the sentence. In the course of analyzing the sentence *People wearing hats is unusual*, a temporary ambiguity exists after the first three words have been analyzed (in two ways). This ambiguity is resolved when the word *is* comes on the scene. When one proceeds from left to right, some of the temporary ambiguities are resolved in situ, because the context built up on the left leaves no room for particular choices of word categories or syntactic structures at a given point in the analysis. Further assistance in resolving ambiguities is provided by restrictions. Certain nouns, for example, must be preceded by an article or possessive when they are singular in subject position; e.g., we may have *Book ends are needed*; *The book ends happily*, but not *Book ends happily*. Hence the application of the restriction on these nouns, say, after the first word of the sentence *Book ends are needed* eliminates one analysis immediately (the one which would take *book* as subject and *ends* as verb).

3. The String Program

This section describes the common features of two computer programs which employ the procedure of Section 2 [*45*]. The first, written in IPL V, has been in operation since the fall of 1964, and the second, written in FAP for the 7094, since December 1965.[5]

[5]The IPL V program was written by James Morris and the FAP program by Carol Raze.

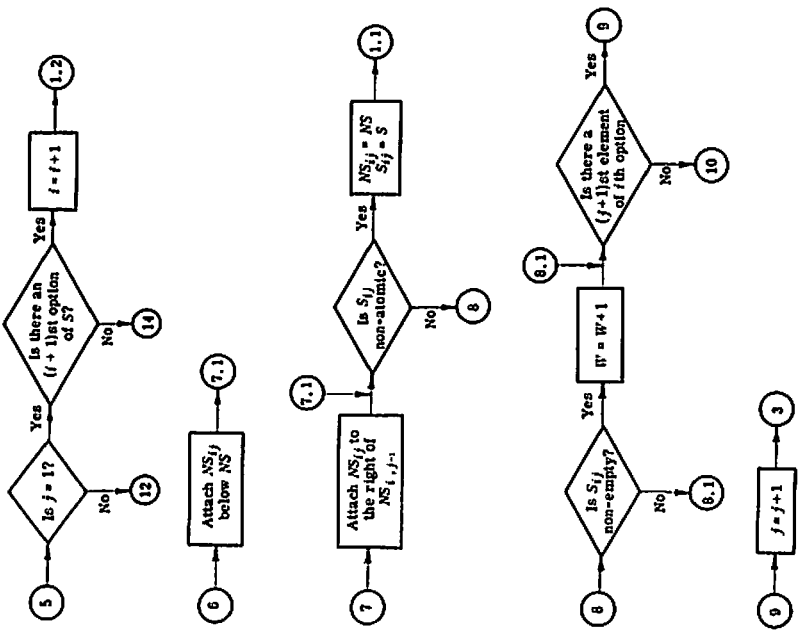
W = 1
NS = 1st node
S = 1st string
i = 1
j = 1
Look at
i'th option
of S
Is there
an R_i?
Execute R_i
Was R_i
successful?
Look at j'th
element of
option i
Is S_{ij}
atomic?
Is S_{ij}
non-empty?
Look at the
category list
of W
Is there an
S_{ij} on it?
Is j = 1?
Yes
No

FIG. 1a

Is there an
(i + 1)st option
of S?
Is j = 1?
Attach NS_{ij}
below NS
Is S_{ij}
non-atomic?
Attach NS_{ij} to
the right of
$NS_{i,j-1}$
NS_{ij} = NS
S_{ij} = S
Is there a
(j+1)st element
of i'th option
Is S_{ij}
non-empty?
W = W+1
j = j+1
Yes
No

FIG. 1b

FIG. 1. Building the analysis tree. S = string being processed. S_{ij} = jth element of ith option of S. NS = node corresponding to S. W = current word to be analysed. R_i = restriction on ith option of S.

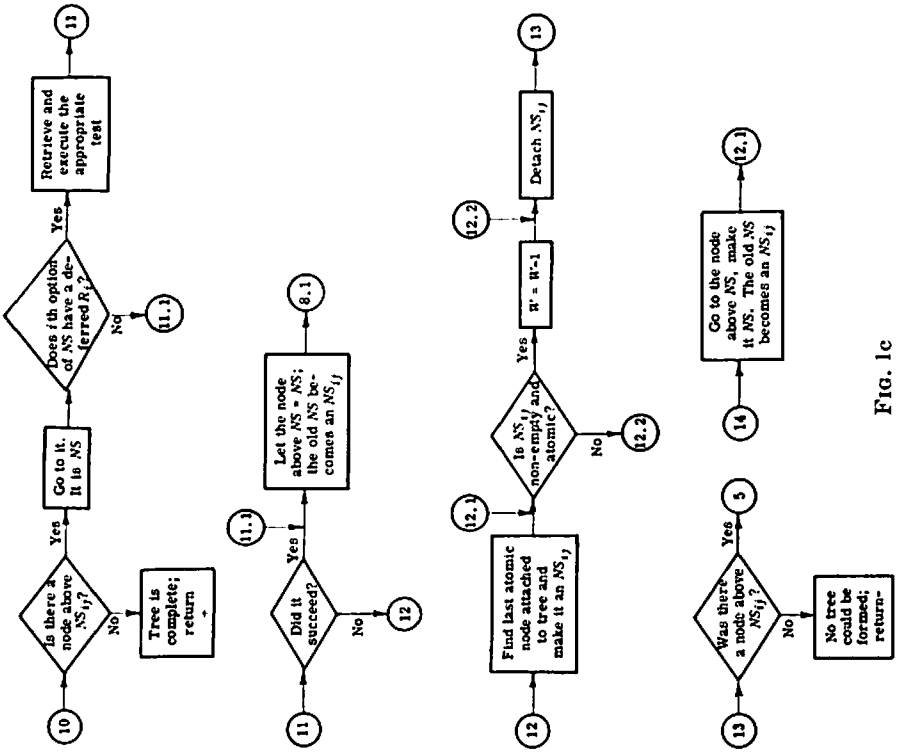

FIG. 1c

3.1 General Description

The analysis routine of the program is a fairly general language-independent processor which when supplied a grammar (coherent set of definitions) and an input string (sentence) to parse, proceeds from the beginning to the end of the sentence trying to analyze the sentence as a case of some chain of substitutions contained in the definitions. At each decision point, it takes the first available option, backing up to try successive options either because the chosen path fails or in order to obtain alternative analyses after the first successful analysis. The record of the analysis is kept (internally) in the form of a tree. The steps in building the analysis tree in the FAP program are shown in Fig. 1.[6]

The grammar is entered into the machine as a set of definitions of "grammar strings." A grammar string S is defined to be S_1 *or* S_2 *or* $\cdots$ *or* S_n; S_i is defined to be S_{i1} *and* S_{i2} *and* $\cdots$ *and* S_{im}; S_{ij} is a string similar to S. The S_i's are called the options of S, and the S_{ij} 's the elements of S_i. A chain of substitutions terminates when the S_{ij} 's are atomic, i.e., word-category symbols. [An elementary linguistic string in the sense of Section 2 thus corresponds to an option of a grammar string as defined here.] The machine representation of a grammar string is shown in Fig. 2. What mainly distinguishes the definitions from other formal language specifications are:

(1) The content of the definitions: Each definition represents a linguistic string or set of strings, so that there is a one-to-one correspondence between the linguists' string grammar and the set of definitions accepted by the program.

(2) The restrictions: Associated with each definition is a restriction list which contains tests corresponding to the linguistic restrictions associated with the given string or set of strings. The restrictions (tests) are written in a special language of basic routines. Despite the fact that restrictions can be looked upon theoretically as abbreviations for families of unrestricted strings, it would be cumbersome (though not impossible) [*47*] to write a string grammar of a natural language without them, especially if some degree of refinement is desired.

[6]Figures 1 and 2 are taken from Carol Raze, The FAP String Analysis Program, in [*45*]. The tree structure in the FAP program differs from the illustrated trees of Fig. 3–5 (which use the IPL V form) in that a node *NS* in the FAP tree contains 2 pointers which point, respectively, to the left sibling, or if *NS* is left-most, to the parent node, and to the right sibling if there is one. The node *NS* (representing the grammar string S) also has a grammar pointer to the string S in the grammar.

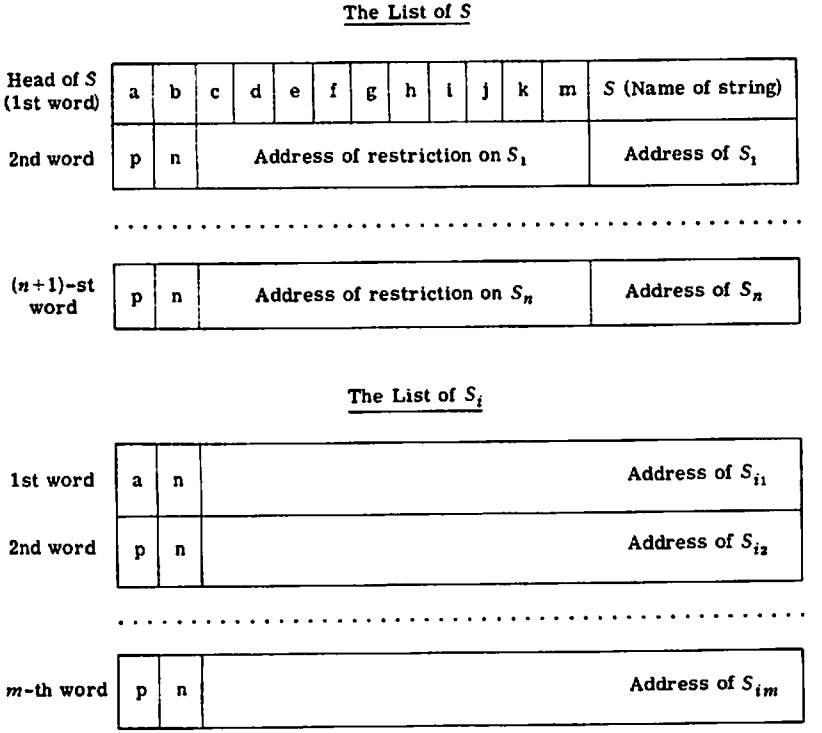

FIG. 2. Machine representation of a string S. S is defined to be S_1 *or* S_2 *or* $\cdots S_n$. S_i is defined to be S_{i1} *and* S_{i2} *and* $\cdots S_{im}$, S_{ij} is a string similar to S. The alphabetical symbols represent fields. Letters c through m are fields consisting of one bit reserved for string properties and are found only in the head of a string. If a certain property is present the bit in the appropriate field will be set to one(1). The first two fields of a word in any list (a, b and p, n) are necessary for the program. The fields represent:

- a Always 0—signals the beginning of a list
- b Always 1—signal for a head
- c Atomic property (if present, the string has only a head)
- d Empty atomic (null of adjunct sets)
- e Special empty atomic (null of omission and zeroing)
- f Transparent
- g Save
- h Freeze
- i Recursive
- j Min-word
- k Special process
- m Repetitive property
- n Always 0—signal for a nonhead
- p Always 1—signal for the middle of a list

(3) Special process definitions: Various words in the language (e.g., coordinate and comparative conjunctions, comma, parentheses) are marked "special," where special means that a word carries its own grammar definition as part of its dictionary entry. When a special word

becomes the current sentence word the dictionary-carried definition is inserted into the current string in a prescribed position. Thus certain structures (headed by special words) need not be predicted by the otherwise deterministic procedure; they are allowed to interrupt anywhere and carry their own definitions. These definitions may call on strings in the grammar, or alternatively the strings can be composed at run-time using an R1-specify restriction (described below). The definition for *and*, for example, carries an R1 restriction which retrieves a portion of the analysis tree constructed prior to the appearance of *and*, and constructs a set of strings in isomorphic correspondence to the retrieved structure, i.e., carries out what was called structural repetition in the preceding description of conjunctions.

3.2 Restrictions

In the program, a restriction is a test which must be executed successfully before a particular sequence of sentence words can be accepted as an instance of the grammatical entity (definition) which carries the restriction. Thus all restrictions are basically wellformedness tests. In practice, however, it is possible to perform some of these tests before any attempt is made to determine whether the definition in question can be satisfied starting with the current sentence word. In addition, therefore, to the wellformedness tests proper (R2 in the set of restriction types below) which are performed after a definition φ is associated with particular words in the sentence, there are specialized restriction types (R1, R3–R7) which are executed before φ is attempted:

R1 (specify)	Replaces the given definition of φ with one obtained by executing the R1 test.
R2 (wellformedness)	Checks that the instance of φ just constucted meets detailed wellformedness tests.
R3 (disqualify)	Disallows a particular value (option) of φ which it will be impossible to satisfy with the words in the given sentence, or in the analysis thusfar constructed.
R4 (omission)	Accepts the null element of omission as value of φ in an omitting *wh* string, or the null element of zeroing in a conjunctional string which allows zeroing.
R5 (SP scope)	Tests whether a special node φ (corresponding to a special process definition) may be tried on the next higher level of the analysis tree.

R6 (SP insert)	Tests whether a special node φ may be inserted at the present point in the current string.
R7 (category check)	Disqualifies φ if the current sentence word is not in a word category which can begin φ.

All restrictions are composed from a basic vocabulary of restriction routines. These are mainly divided into logical tests and routines for moving about the analysis tree (location routines). An important feature of the location routines is "transparency," a device which was introduced to compensate for certain discrepancies between the linguistic string grammar and its machine representation. For example, in the interests of computational simplicity, appearances of X in string definitions are replaced by $\bar{X}$, where $\bar{X} = l_X\ X\ r_X$, and l_X and r_X are allowed to take on the value zero. As a result of this convention, the analysis tree contains nodes which are not mentioned in the linguistic description of the same structures and interfere with a simple statement of restrictions as relations among elements of the same string or of host-adjunct pairs. To overcome this obstacle, various routines which search the analysis tree "see" only the essential linguistic nodes; other nodes are passed through in the course of the search as though they were transparent. [The nodes which are to be considered transparent are specified by the user.]

To illustrate, suppose the simple grammar of Section 1 were to be entered into the machine as the following definitions:

z_1 (center) $= \bar{N}_1\ tV\ \bar{N}_2$	[subscript indicates position in string]
$\bar{X} = l_X\ X\ r_X$	[for $X = N, tV, P$]
$l_X = l_{tV} = r_{tV} = l_P = r_P = 0$	[i.e., all adjunct sets except r_N are empty]
$r_N = PN$ string or 0	[i.e., r_N has as value a PN string or zero]
PN string $= \overline{P}\ \bar{N}$	
Transparent nodes: $\bar{X}, r_X, l_X$	[for $X = N, tV, P$]

The analysis tree corresponding to the string analysis of the sentence *Cars without brakes cause accidents* would be as shown in Fig. 3. In the course of checking *cars* against *cause* for agreement in number, the restriction routines will pass through the transparent nodes $\overline{tV}$ and $\bar{N}_1$.

Transparency is also used in preparing the output because the tree representation somewhat obscures the fact that a sentence has been decomposed into elementary strings. The string character of the analysis is restored in the output display by causing the print routine to

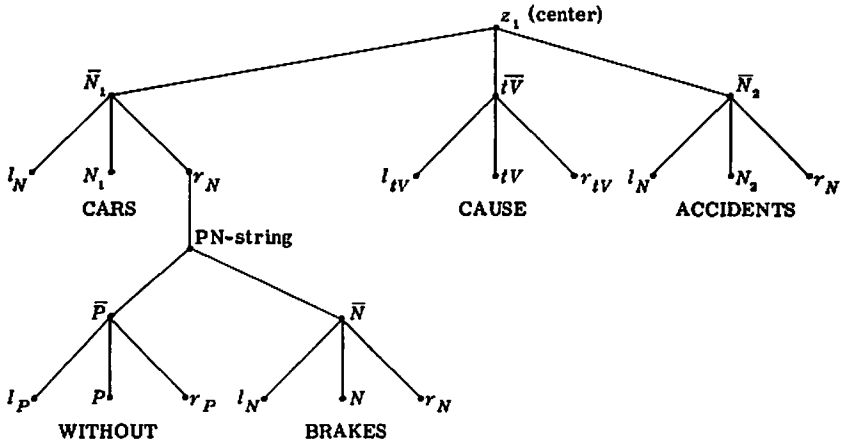

FIG. 3

ignore transparent nodes. Each nontransparent nonatomic node (i.e., each linguistic string) is assigned a numbered line of output on which is written the values of nontransparent nodes which appear under it in the tree: For an atom, the value is a sentence word; for a linguistic string, the value is the number of the output line on which the string is written. The output corresponding to the analysis tree above would be:

(1) z_1 (center) = CARS 2. CAUSE ACCIDENTS
(2) *PN* string = WITHOUT BRAKES

The fact that the number 2. falls to the right of *cars* indicates that the *PN* string on line (2) is a right adjunct of *cars*.

3.3 Special Processes

When a word marked "special" (i.e., carrying an *M* attribute) becomes current, the program inserts an element *Mj* (= the value of the *M* attribute for the given word) into the current string ξ at a point just following the most recently satisfied element X_l. *Mj* is the name of a grammar definition (*M*1 for *and*, *M*2 for *or*, etc.) and unless the insertion of *Mj* is disqualified by an R6 test in *Mj*, the various options of *Mj* are tried just as though *Mj* had been the $(l + 1)$th element of ξ. If all the options of *Mj* fail, the program erases *Mj* and tries X_{l+1} in the normal manner. It also removes the "special" tag from the current word since X_{l+1} might be satisfied by this word in one of its nonspecial category assignments (e.g., *but* as adverb rather than conjunction in *They found but one*); in this case the program continues from this point as usual.

However, if the program finds itself at the end of the current string without having used up the special word, it restores the "special" tag to the word and ascends one level in the tree (unless blocked by an R5 test). It then repeats the whole process at the new level, beginning with the insertion of an Mj node at the current point in the string at that level. If all efforts (special and nonspecial) fail, the program backs up to the previous sentence word, as it would normally on failure.

This scheme is especially apt for conjunctions. For example, suppose the sentence in Fig. 3 were interrupted by a conjunctional string after the word *brakes*. Figure 4 shows two alternative ways in which the string *and headlights* could be attached to the analysis tree. The program would first conjoin *and headlights* to the deeper string (*without brakes*); it would later obtain the second analysis in which *and headlights* is conjoined to a string (*cars cause accidents*) higher in the tree.

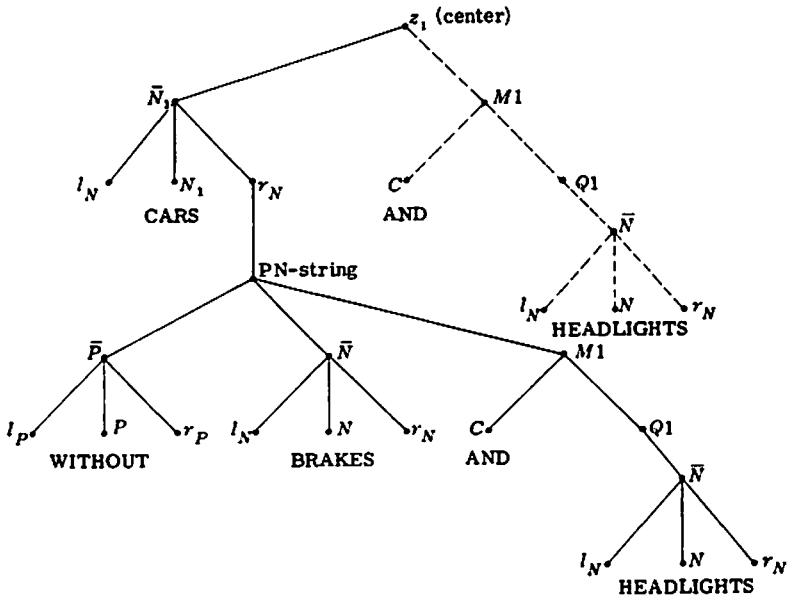

FIG. 4

3.3.1 Coordinate Conjunctions: and, or, but

The Mj definitions for coordinate conjunctions all contain an option $C\,Q1$, where the content of $Q1$ is specified by an R1 restriction as follows: Given that Mj has been inserted following X_l in ξ, the options of $Q1$ are:

$$X_l;\quad X_{l-1}X_l;\quad X_{l-2}X_{l-1}X_l;\quad \cdots;\quad X_1 \cdots X_{l-1}X_l.$$

That is, the restriction carries out the "structural repetition" of Section 1. In Fig. 4, the first option X_1 of $Q1$ is used in both conjoinings shown.

While a gross string analysis of conjunctional sequences need not take account of the existence of zeroed elements, the identifying and supplying of zeroed elements in (or just before) conjunctional strings facilitates applying to a conjunctional string $C\xi_2$ the restrictions housed in the string ξ_1, which apply as well to ξ_2. For example, in the sentence *She likes toys and animals too*, the analysis which would have animals liking toys too (similar in form to *She likes toys and he too*) is ruled out by the restriction as to subject-verb agreement in number applied to the complete ξ_2: ∄ *She likes toys and animals likes toys too.*

An analogous situation exists in the *wh* strings to which the operation of omission has been applied. Here, too, it is often useful to redirect a restriction from a null element (here the omitted element of an adjunct *wh* string) to the appropriate element of the host string. For example, in sentence 5 of the sample text, the program checks *was* (line 7) against *porosity* (line 2) for agreement in number.

3.3.2 Scope-Marked Conjunctions: Either···or, neither···nor, both···and

In terms of the operation of structural repetition by which conjunctional strings are obtained, the words *either* (as part of *either-or*), *neither* (as part of *neither-nor*) and *both* (as part of *both-and*) can be seen to mark the point in the host string beyond which elements cannot be "repeated" in the conjunctional string, e.g., ∃ *John wrote a paper and Mary corrected it*, ∃ *John both wrote and corrected a paper*, ∄ *John both wrote and Mary corrected a paper.* That is, a pair $C' \cdots C$ (e.g., *either* ··· *or*) marks off a structure X (a string or string segment) which also appears following C in the sentence: $C'\ X\ C\ X$. Since X is the expected structure when C' occurs, it is possible to define a special process, initiated by C', which inserts the string $C'\ X\ C$ at the interrupt point; when the $C'\ X\ C$ string is satisfied, the program reverts to its normal operation and finds an X as it expected before the interruption. The tree in Fig. 5 illustrates the result of this process. Here C (*or*) is a non-special element, because it is not an independent string head, but part of $C'\ X\ C$.

3.3.3 Comma

The comma is troublesome because in some of its uses it is a required string element (e.g., in conjunctional sequences of the form X, X *and* X)

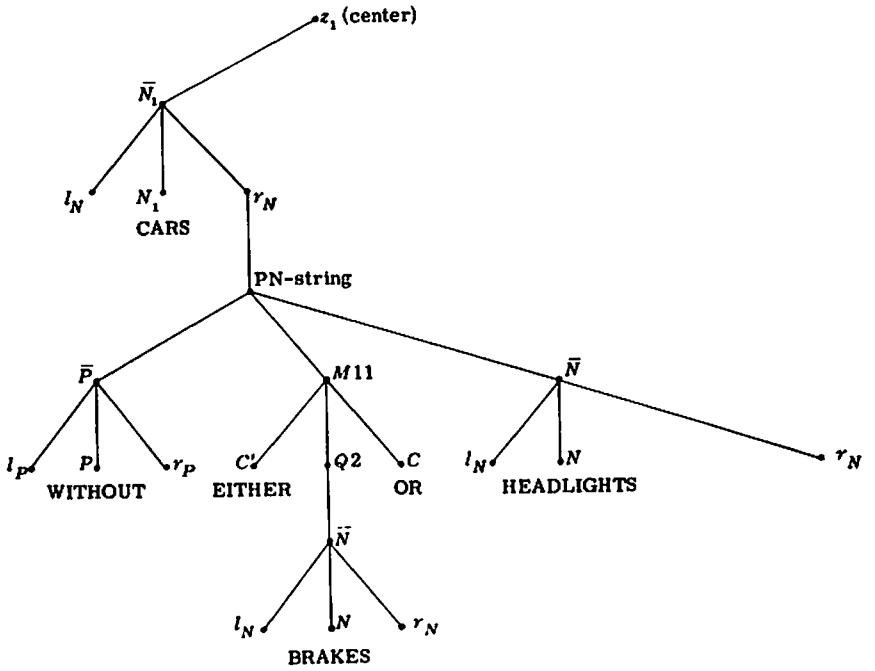

FIG. 5

while in other uses it is punctuational and as such not entirely subject to grammatical description. In order to proceed, we assume that if a given comma does not analyze as a conjunction ($Q1$ option with various restrictions), then it is all right to accept the comma as punctuation. The words following the comma must then be analyzed by the ordinary grammar. In addition, particular strings which either should or should not be set off by punctuation commas carry their own restrictions to this effect, e.g., *The children asleep, they* ···

3.3.4 Comparative Conjunctions: *–er ··· than, as ··· as*

The comparative forms in English are particularly rich structurally and one way to handle them is to see their resemblance to strings and processes already defined. Accordingly, the special process definition which is initiated by *than* (when preceded by *more, less* or *X-er*; *X* = adjective, a few adverbs) or by *as* (when preceded by adverbial *as*) contains several options:

(1) Conjunctional. Compare, for example,

apples and oranges	*more apples than oranges*
People will buy the book and read it.	*More people will buy the book than read it.*
	People will more (often) buy a book than read it.
	but ∄ *People buy more books than read them.*

With a few restrictions which depend on the placing of the comparative marker in the host string, the large body of comparative strings can be obtained using the same set of strings ($Q1$) as are generated in the conjunctional special process.

(2) *wh* like. There exists in the grammar a *wh* string with omission which adjoins a whole sentence, e.g., *The crowd dispersed quietly, which they had not expected.* When the ultimate verb in this string is such as can occur with *that S* as its object (S = sentence), there is a similar comparative string:

The crowd dispersed quietly, which they had not expected.	*The crowd dispersed more quietly than they had expected.*

The *wh* string which adjoins an N also has a comparative analog:

The boxes which we asked for have arrived.	*More boxes than we asked for have arrived.*

Than and *as* can therefore be placed in the same class as *which* (as heading the same strings) and handled by the ordinary grammar, with the restriction that *than* and *as* each be preceded in the sentence by the appropriate comparative marker, and with certain restrictions on the strings headed by *than* and *as* vs. *which*. There is also a shortened form of the passive strings of this type (headed only by the comparative string heads); e.g., *More people than expected were there, The crowd dispersed more quietly than expected*, etc.

(3) *N is A* strings. As a right adjunct of N, a comparative marked adjective behaves (in string terms) as though it were preceded not by N alone but by *N is A* (A = adjective), from which the various $Q1$ strings can be obtained:

Someone smarter than I could solve the problem.
Someone smarter than I am could solve the problem.
Someone smarter than I am patient could solve the problem.

(4) Inverted assertion: $\Sigma\, t\, V\, \Omega\, C\, t\, \Sigma$. This option covers the comparative strings in which the subject is permuted with W, *have*, *be*, or *do*. For example, (from a text):

> *Mammals have generally been conceded to possess a much wider "plasticity" and "adaptability" of coordination than do amphibians.*

(5) Special zeroing, for small adverb subclasses: *more beautiful than ever, more people than ever.*

3.4 Treatment of Syntactic Ambiguity

3.4.1 Saving Reusable Portions of the Analysis

Even an ambiguous sentence is ambiguous only in certain respects. It is likely, then, that two (or more) syntactic analyses of a given sentence will be identical in their analysis over some portion or portions of the sentence. It would be wasteful to recompute such portions for each analysis. Also, in the course of obtaining an analysis, numerous false paths are tried; i.e., the first m words of an n-word sentence may be subject to an analysis which is later found not to fit in an analysis of the entire n-word sequence. In this case, too, it would be wasteful to recompute common portions of the various paths. The string analysis programs therefore are equipped with a device called "saving" by which computed instances of certain strings (substructures of the analysis tree) are copied before being dismantled. These substructures are then available for reuse as "plug-in" units in other analyses or paths.

A structure which contains no restriction that examines the context outside that structure is called *autonomous*, and can be saved and used in any context which calls for such a structure—subject, of course, to any further restrictions imposed by the new context. The grammar writer can minimize computer time by making as many strings as possible autonomous; but it should be noted that this aim conflicts with the desire to house restrictions as low in the tree structure as possible, so that non-wellformed sequences will be discarded soon after their construction. To make it possible to save a particular instance φ of a nonautonomous structure, when this is justified, the program keeps track of the paths followed in executing restrictions during the construction of φ; and (in the IPL program) φ is saved only if no restriction path extended beyond φ (to a node not in the substructure of φ). Making use of this possibility, many restrictions are in the form of an

implication in which the if-clause checks for some feature which separates the autonomous from the nonautonomous cases. The FAP program also saves that portion of a nonautonomous φ which was constructed before a restriction path extended beyond φ.[7]

3.4.2 *Permanent Predictable Ambiguities*

By far the largest number of alternative analyses are of the type in which the breakdown of the sentence into elementary strings in the various analyses is the same but the string assignment is different. The most frequent situations are these:

(1) Alternative analyses due to nesting vs. repeatability: The operation of adjoining, say, a right-adjunct of N to a particular N is repeatable (*a man from Philadelphia whom I met*). In many cases, such as in a sequence of $P\ N$'s in r_N: $N_1\ P\ N_2\ P\ N_3 \cdots$, the second adjunct can adjoin either N_1 (repeatability) or N_2 (nesting): *the house of my friend in Philadelphia* $\cdots$.

(2) Same string in different adjunct sets which enter at the same point: Again to use the example of a $P\ N$ string, the $P\ N$ string is a member of the sets r_N, r_V, and s.a. (sentence adjuncts). There are certain points in a sentence at which two or all three of these sets can enter; e.g., all three can enter after the object, when the object is N:

r_N:	*He opened the box with the blue cover.*
rV:	*He opened the box with interest.*
s.a.:	*He opened the box with time to spare.*

In some cases, we can resolve the ambiguity by bringing a restriction to bear; e.g., for $P\ N$ in r_V, *of* is not an acceptable value of P.

Until we can establish subclasses that will give us better discrimination, we have permanent predictable ambiguities in these situations, on which it seemed unrewarding to spend computer time. Where there are alternative analyses due to repeatability vs. nesting, the program chooses the analysis due to nesting. Where strings in r_N, r_V, and *s.a.* overlap, the program assigns the common string to the first adjunct set having permission to enter in the given situation.[8]

[7]This technique of partial saving is described more fully in Carol Raze, "User's Manual for the FAP String Program," in preparation.

[8]It should be emphasized that not printing the permanent predictable ambiguities is not a solution to the problem of obtaining the intended (preferred) analysis; this lies in more refined subclasses and other constraints which may be drawn from the textual context. Nevertheless, at this stage of research, there is a considerable practical gain in being able to suppress ambiguities selectively.

The min-word test, which accomplishes this assignment, is based on a simple observation: If in the course of back-tracking through an n-word sentence in order to obtain an alternative analysis, the program has reached back as far as the mth word ($m \leq n$) and no further (i.e., it has destroyed the given analysis for words m through n), then in starting forward from this point any attempt to construct an adjunct string α_i beginning with word m will only result in an ambiguity if any previous analysis contains α_i as an adjunct string beginning with word m. The program keeps a record of the current value of "the min-word" (m in the above observation). There also exists a "suppress list" containing the names of the strings $\alpha_1, \alpha_2, \cdots$, etc., which we do not wish to have the program shift from one adjunct slot to another in order to obtain different analyses. After each analysis, the program scans the entire analysis tree comparing the name of each node with the names of the strings on the suppress list and in case of equality attaches the attribute pair N_i, N_j to the first word subsumed under the node in question; $N_i = \alpha_i$, the name of the string at that node, $N_j =$ the first node above N_i, which in the machine representation of the grammar is the name of the adjunct set from which α_i was taken in this analysis (if α_i is occurring as an adjunct). The following R3-restriction on each α_i prevents the program from taking α_i as an adjunct somewhere else in the tree: Not all the tests (a), (b), and (c) are true: (a) There has already been at least one successful analysis; (b) the current word is the min-word; (c) N_i, N_j appear as an attribute pair of the current word.

3.4.3 *Freezing Grouped Segments*

While the min-word test deals with the external relations of the strings obtained in a decomposition of a sentence, the freeze test concerns their internal structure. If it can be established that the words of an already computed structure φ have no other possible segmentation in the given sentence, then in undoing an analysis it is justified for the program to treat φ as though it were a single word or a "frozen" block of words. The freeze test is executed when the program is about to dismantle a computed structure φ in the course of back-tracking. If the test succeeds, the program skips back over all the words subsumed by φ and continues its back-tracking from there; i.e., it detaches the node φ (with all its substructures) from the analysis tree as a unit.

As an example, consider the distinguished segment $l_N N$ (N with its left adjuncts). If the sentence word satisfying N (the "core value" of $l_N N$) is the type of noun which requires a preceding article or possessive, then except under statable conditions the sequence of words which stretches from a preceding article or possessive up to and including the

noun in question cannot be differently segmented.[9] Thus in the sentence

A 10–40 per cent w/r sucrose density gradient was prepared according to the method of B. and L

the word sequence *a 10–40 per cent w/r sucrose density gradient*, as an instance of $l_N N$, can be "frozen." The freeze test is only applied after one analysis of the sentence has already been obtained.

3.5 Output

As illustrated in Section 2.2, in an analysis of a sentence each component linguistic string is assigned a numbered line of output on which are printed the words of the string in the sentence. If an element in a string S_1 is itself a string S_2, then the number of the output line of S_2 appears in the element position of S_2 in S_1. Sentence adjunct strings are preceded by an asterisk, *, to distinguish them from all other adjuncts, which are positionally indicated: If a right (left) adjunct string S_2

04 Ferrous Extraction and Refining

M04--24522. THE EFFECT OF CALCIUM CARBONATE ON THE REDUCIBILITY OF IRON-OXIDE AGGLOMERATES. P.K. Strangway and H.U. Ross. Can Met Quart, v 4, no 1, Jan-Mar. 1965, p 97-111.

Briquettes, consisting of pure ferric oxide and ferric oxide with 1, 2, 5 and 10% calcium carbonate, were sintered at 1200 C. They were then reduced by hydrogen in a loss-in-weight furnace at temperatures ranging from 600 to 1000 C. It was found that calcium carbonate increased the reducibility in all instances. At low reduction temperatures, the effect was more pronounced as the calcium carbonate content increased. This, in turn, corresponded to a greater initial porosity which was developed during sintering. At higher reduction temperatures, however, this effect was more pronounced for briquettes with small calcium carbonate additions. In this instance, the porosity which was developed during reduction became more important than that which was developed during sintering. 22 ref. (AA)

FIG. 6

[9]The statable exceptions mentioned above involve reanalyzing the N as a left adjunct of N, e.g., as part of a compound noun or compound adjective.

adjoins an element E of S_1, the number of the output line of S_2 is printed to the right (left) of E in S_1.

Figures 7–14 contain the computer outputs for the sentences of the metallurgy abstract shown in Fig. 6. The second parse of Sentence 4 (Fig. 11), if it exists at all for this choice of words for the strings in question, would have to be construed as something like: *The effect was more pronounced as (measured by) the increased calcium carbonate content.* A more detailed form of output, giving the grammatical names of every string element, is shown in Fig. 11.

```
SENTENCE  1. BRIQUETTES , CONSISTING OF PURE FERRIC OXIDE AND FERRIC
            OXIDE WITH 1 , 2 , 5 AND 10 PERCENT CALCIUM CARBONATE , WERE
            SINTERED AT 1200 DEGREES CENTIGRADE .

     PARSE     01
 1. SENTENCE          =              2.     .

 2. C1 ASSERTION      =    BRIQUETTES ,  3.    WERE    SINTERED  4.

 3. VING +            = CONSISTING    OF  5. OXIDE AND  6.

 4. C20 P N           =       AT            7. DEGREES 10.

 5.  L-A OF N         =                      PURE FERRIC

 6. CONJ STG          = 11. OXICE 12.

 7.  L-A OF N         =          1200

10. ADJ IN R-N        = CENTIGRADE

11.  L-A OF N         =                      FERRIC

12. C20 P N           =       WITH         13. CARBONATE ,

13.  L-A OF N         =                      1 , 14. PERCENT          CALCIUM

14. CONJ STG          = 2   , 15.

15. CONJ STG          = 5   AND 16.

16. CONJ STG          = 10
NO MORE PARSES
```

FIG. 7

```
SENTENCE  2. THEY WERE THEN REDUCED BY HYDROGEN IN A LOSS-IN-WEIGHT FURNACE
            AT TEMPERATURES RANGING FROM 600 TO 1000 DEGREES CENTIGRADE
            .

     PARSE     01
 1. SENTENCE          =              2.

 2. C1 ASSERTION      =    THEY       WERE * 3.   REDUCED  4.

 3. ADVERB            = THEN

 4. C20 P N           =       BY          HYDROGEN  5.

 5. C20 P N           =       IN           6. FURNACE  7.

 6.  L-A OF N         = A                    LOSS-IN-WEIGHT

 7. C20 P N           =       AT          TEMPERATURES 10.

10. VING +            = RANGING    FROM 11. DEGREES 12.

11.  L-A OF N         =          600 TO 13.

12. ADJ IN R-N        = CENTIGRADE

13. CONJ STG          = 1000

NO MORE PARSES
```

FIG. 8

```
SENTENCE  3. IT WAS FOUND THAT CALCIUM CARBONATE INCREASED THE REDUCIBILITY
            IN ALL INSTANCES .
     PARSE     01
 1. SENTENCE          =              2.

 2. C1 ASSERTION      =    IT          WAS      3.

 3. C132 VEN O-PASS   = FOUND    THAT  4.

 4. C1 ASSERTION      =     5. CARBONATE    INCREASED     6. REDUCIBILITY     * 7.

 5.  L-A OF N         =                                      CALCIUM

 6.  L-A OF N         = THE

 7. C20 P N           =      IN          10. INSTANCES

10.  L-A OF N         =          ALL

NO MORE PARSES
```

FIG. 9

```
SENTENCE  4. AT LOW REDUCTION TEMPERATURES , THE EFFECT WAS MORE PRONOUNCED
          AS THE CALCIUM CARBONATE CONTENT INCREASED .
     PARSE     01
 1. SENTENCE          =              2.

 2. C1 ASSERTION      = * 3.    4. EFFECT    WAS      5. PRONOUNCED     * 6.

 3. C20 P N           =       AT            7. TEMPERATURES ,

 4.  L-A OF N         = THE

 5. ADVERB            = MORE

 6. C161 CS1 C        =       AS  10.

 7.  L-A OF N         =                     LOW                REDUCTION

10. C1 ASSERTION      =    11. CONTENT    INCREASED

11.  L-A OF N         = THE                                    CALCIUM CARBONATE
```

FIG. 10

```
SENTENCE  4. AT LOW REDUCTION TEMPERATURES , THE EFFECT WAS MORE PRONOUNCED
          AS THE CALCIUM CARBONATE CONTENT INCREASED .
     PARSE     02
 1. SENTENCE          = INTRODUCER CENTER END MARK
                              2.

 2. C1 ASSERTION      = *     SUBJECT     * VERB  * OBJECT          R-V  *
                      * 3.    4. EFFECT     WAS      5. PRONOUNCED      * 6.

 3. C20 P N           =  L-P PREPOSITION   N
                         AT            7. TEMPERATURES ,

 4.  L-A OF N         = ARTICLE QUANTIFIER ADJECTIVE TYPE NS NOUN
                        THE

 5. ADVERB            = ADVERB
                        MORE

 6. C160 CSO O-BE     = L-CS CSO OBJ-BE          *
                           AS  10. CONTENT 11.

 7.  L-A OF N         = ARTICLE QUANTIFIER ADJECTIVE TYPE NS NOUN
                                            LOW              REDUCTION

10.  L-A OF N         = ARTICLE QUANTIFIER ADJECTIVE TYPE NS NOUN
                        THE                                  CALCIUM CARBONATE

11. C132 VEN O-PASS   =  VEN       * PASSIVE-O R-V  *
                         INCREASED

NO MORE PARSES
```

FIG. 11

```
SENTENCE  5. THIS , IN TURN , CORRESPONDED TO A GREATER INITIAL POROSITY
          WHICH WAS DEVELOPED DURING SINTERING .
     PARSE     01
 1. SENTENCE        =              2.    .

 2. C1 ASSERTION    =      3.     * 4.   CORRESPONDED    TO  5. POROSITY  6.

 3.  L-A OF N       = THIS ,

 4. C20 P N         =       IN           TURN ,

 5.  L-A OF N       = A                    GREATER INITIAL

 6. C85 WH STG      = WHICH  7.

 7. C1 ASSERTION    =      ()        WAS      DEVELOPED      *10.

10. C164 CS4 SN     =       DURING SINTERING

NO MORE PARSES
```

FIG. 12

```
SENTENCE  6. AT HIGHER REDUCTION TEMPERATURES , HOWEVER , THIS EFFECT
          WAS MORE PRONOUNCED FOR BRIQUETTES WITH SMALL CALCIUM CARBONATE
          ADDITIONS .
     PARSE     01
 1. SENTENCE        =              2.    .

 2. C1 ASSERTION    = * 3. HOWEVER ,    4. EFFECT    WAS      5. PRONOUNCED  6.

 3. C20 P N         =       AT           7. TEMPERATURES ,

 4.  L-A OF N       = THIS

 5. ADVERB          = MORE

 6. C20 P N         =       FOR          BRIQUETTES 10.

 7.  L-A OF N       =                      HIGHER             REDUCTION

10. C20 P N         =       WITH         11. ADDITIONS

11.  L-A OF N       =                      SMALL              CALCIUM CARBONATE

NO MORE PARSES
```

FIG. 13

```
SENTENCE  7. IN THIS INSTANCE , THE POROSITY WHICH WAS DEVELOPED DURING
          REDUCTION BECAME MORE IMPORTANT THAN THAT WHICH WAS DEVELOPED
          DURING SINTERING .

     PARSE     01
 1. SENTENCE        =              2.    THAN  3.    .

 2. C1 ASSERTION    = . 4.    5. POROSITY  6.    BECAME     7. IMPORTANT

 3. CONJ STG        = 10.

 4. C20 P N         =      IN          11. INSTANCE ,

 5.  L-A OF N       = THE

 6. C85 WH STG      = WHICH 12.

 7. ADVERB          = MORE

10. C1 ASSERTION    =    13. 14.    (BECAME )    ( IMPORTANT )

11.  L-A OF N       = THIS

12. C1 ASSERTION    =     ()         WAS      DEVELOPED 15.

13.  L-A OF N       = THAT

14. C85 WH STG      = WHICH 16.

15. C20 P N         =      DURING      REDUCTION

16. C1 ASSERTION    =     ()         WAS      DEVELOPED     .17.

17. C164 CS4 SN     =      DURING SINTERING

NO MORE PARSES
```

FIG. 14

3.6 Discussion

3.6.1 Natural Form for Discontinuous Entities

The method of immediate constituent analysis is powerful up to a certain point because to a very large extent the modifiers of a word will be found next to it in the sentence. However, it has two drawbacks: One, there are cases of modifiers which are at a distance from the word they modify, e.g., adverbs at a distance from their verb, as in *Softly she tiptoed out of the room*; or a relative clause at a distance from its

noun, as in *An explanation is here offered which is adequate for all the cases above*. Two, more seriously, there are a great many grammatical dependences between words which are in different constituents, as in number agreement between the subject noun and its verb, or the requirement for an anticipatory *it* subject in sentences like *It is surprising that he was there* where *that he was there* requires that the subject be *it*.

The problem of dependence at a distance as in the above examples is also of interest for computational work because no limit can be specified for the distance between the words or clauses which participate in the dependence: e.g., *It might perhaps begin to seem to · · · to become surprising to John and Mary and Jim and · · · and Jack that she was there*. This problem exists when the sentence is taken as constructed from words or constituents. However, there is one segmentation of sentences in which the problem of dependences at a distance does not arise. This is the segmentation of each sentence into a center string and adjunct strings. Every grammatical dependence in a sentence occurs between parts of a string or between an adjunct and the string to which it is adjoined.[10] For example, the number agreement is between the subject and verb of a string, e.g., between *explanation* and *is* in the center string *An explanation is here offered* in the sentence *An explanation is here offered which is adequate for the cases above*; and *which is adequate for all the cases above* may indeed be at a distance from *explanation*, which it modifies, but is adjoined to the string *An explanation is here offered*, of which *explanation* is a part. This string analysis can be carried out in a regular way on all sentences of a language, and because it is a form of grammatical analysis in which no dependences are at a distance, it is particularly convenient for syntactic computation.

3.6.2 Computational Economy of Strings

By recognizing a word (as member of a word class) in the sentence as being a word in a string in that sentence, word-by-word string analysis can use rules for string succession in sentences, instead of rules for word class or constituent succession in sentences. The number of grammatical strings which is adequate for describing the great bulk of the sentences of English is about 125, grouped into about 15 classes, and the only combination rules in string analysis are those that state which class of strings enters strings of some class of strings at some point of the (latter) string. Hence the number of rules for string expectation is

[10] A special case is a relation between a pair of adjuncts adjoined to the same string. An extreme case of this is the zeroing after *don't* in *People who smoke distrust people who don't* where *who smoke* and *who don't* is a system of two adjuncts adjoined to *People distrust people*.

quite small (about 150 rules, i.e., strings and string-class definitions, plus about 150 restrictions in the present computer grammar).

By using strings we also get a framework for expressing restrictions, which can be formulated in respect to a string. The value, to a grammar or analyzer, of having a system in respect to which restrictions can be stated lies in the following: Natural language notoriously has a vast amount of detail and many at least apparent exceptions to every rule. This has not precluded the formulation of precise and detailed grammars of languages. The reason is that the variety of structures in a language are grouped into sets of structures which have a family resemblance plus various specific differences for various subsets of words. With regard to the exceptions, the great bulk of them can be shown to be extensions of a rule to some words beyond the range for which the rule had been defined. For this reason it is most efficient to recognize genera of structures and specific differences (for specific words or word subclasses) within each genus. Using a grammar whose elements are strings grouped into classes makes it possible to state general rules about whole classes of strings, restrictions upon these rules to account for differences in particular strings, and finally, if we wish, further restrictions to account for stylistic and near-idiomatic forms involving very small classes of words. Using string analysis, considerable refinement is thus possible without an undue increase in the number of rules.

3.6.3 Relation to Transformations

Transformational analysis decomposes each given sentence into one or more elementary (kernel) sentences and unary or binary transformational operations upon these. The elementary sentences are assertions, and the given sentence can be looked upon as being transformationally composed from these elementary assertions. The strings into which a given sentence is decomposed by string theory can to a large extent be identified with the elementary sentences of that sentence under transformational theory.

In most cases the relation is direct; e.g., in most cases, the center string is the main kernel sentence of the given sentence, and very many adjuncts can be filled out to the corresponding kernel sentence merely by adding to the adjunct a distinguished word from the string to which the adjunct has been adjoined; which word this is can be recognized directly from the form of the string. For example, in *An explanation · · · which is adequate · · ·* the string *which is adequate* can be filled out to a connective *wh* plus *An explanation is adequate* (these being the transformation and the kernel sentence under transformational analysis).

In other cases the relation between string and transformational decompositions is more complicated. However, here too, the more detailed relations discovered by transformational analysis apply within string-related segments of the sentence. Some of these relations can be incorporated as refinements of the string grammar, but others remain beyond the reach of a practicable string program. However, this does not mean that the string output for the sentences containing these transformational relations is incorrect, but that it misses connections of the given sentence to other sentences of the language.

ACKNOWLEDGMENTS

The string grammar of English used by both programs was tested by Morris Salkoff and the author at the Courant Institute of Mathematical Sciences, New York University. We wish to thank C.I.M.S. for the hospitality of the computer center and for useful advice concerning the structure and preparation of the FAP program. Some of the design of the FAP program was suggested by analogy with methods used in the NU-SPEAK list processing language developed at C.I.M.S. by Professor J. Schwartz.

The author wishes to thank Morris Salkoff for helpful criticism of this manuscript.

REFERENCES

1. Alt., F. L., and Rhodes, I., The hindsight technique in machine translation of natural languages. *J. Res. Nat. Bur. Std.* **B66**, No. 2, 47–51 (1962).
2. Bar-Hillel, Y., A quasi-arithmetical notation for syntactic description. *Language* **29**, No. 1, 47–48 (1953).
3. Bloomfield, L., *Language*. Holt, New York, 1933.
4. Bobrow, D. G., Syntactic analysis of English by computer—a survey. *AFIPS Conf. Proc., Spartan, Baltimore*, Vol. 24, pp. 265–387 (1963).
5. Bolinger, D. L., *Interrogative Structures of American English*, Publ. of the American Dialect Society, No. 28. Univ. of Alabama Press, University, Alabama, 1957.
6. Brooker, R. A., and Morris, D., An assembly program for a phrase structure language. *Computer J.* **3**, No. 3, 168–174 (1960).
7. Bross, I., A syntactic formula for English sentences: Application to scientific narrative, Ms., Roswell Park Memorial Inst., Buffalo, New York, 1965.
8. Chomsky, N., A transformational approach to syntax. *Proc. 3rd Texas Conf. Probl. Linguistic Anal. English, 1958*, pp. 124–58, Univ. of Texas, Austin, Texas (1962).
9. Chomsky, N., *Syntactic Structures*. Mouton, The Hague, 1957.
10. Chomsky, N., Three models for the description of language. *IRE, Trans. Inform. Theory* **2**, 113–124 (1956).
11. Dougherty, Ching-Yi, and Martin, S. E., Chinese syntactic rules for machine translation, Project for Machine Translation, Univ. of California, Berkeley, California, 1964.
12. Green, B. F., Jr., Wolf, A. K., Chomsky, C., and Laughery, K., Baseball: An automatic question answering, in *Computers and Thought* (E. A. Figenbaum and J. Feldman, eds.), pp. 207–216. McGraw-Hill, New York, 1963.
13. Gross, M., On the equivalence of models of language used in the fields of mechanical translation and information retrieval. Paper presented at the

NATO Advanced Study Institute on Automatic Translation of Languages, Venice, July 1962.

14. Harman, G. H., and Yngve, V. H., Generative grammars without transformation rules. MIT TLEQPR No. 68, Mass. Inst. Technol., Cambridge, Mass., 1962.

15. Harris, Z. S., A cycling cancellation-automaton for sentence well-formedness. *Bull. Intern. Computation Centre* **5**, 75–100 (1966).

16. Harris, Z. S., Co-occurrence and transformation in linguistic structure. *Language* **33**, 283–340 (1957).

17. Harris, Z. S., *Methods in Structural Linguistics.* Univ. of Chicago Press, Chicago, Illinois, 1951.

18. Harris, Z. S., *String Analysis of Sentence Structure*, Papers on Formal Linguistics, No. 1. Mouton, The Hague, 1962.

19. Harris, Z. S., Transformational theory. *Language* **41**, 363–401 (1965).

20. Harris, Z. S., *et al.*, *Transformations and Discourse Analysis Papers* (*T.D.A.P.*) Nos. 15–19. Dept. Linguistics, Univ. of Pennsylvania, 1959.

21. Hays, D. G., Basic principles and technical variation in sentence-structure determination, RAND Paper 1984. Rand Corp., 1960.

22. Hays, D. G., Automatic language-data processing, in *Computer Applications in the Behavioral Sciences* (H. Borko, ed.), pp. 394–421. Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

23. Hays, D. G., Grouping and dependency theories, RAND Paper 1910, Rand Corp., 1960.

24. Hiż, H., Congrammaticality, batteries of transformations and grammatical categories. *Proc. Symp. Appl. Math.* **12**, 43–50 (1964).

25. Hiż, H., Steps toward grammatical recognition. *Intern. Conf. Std. Common Language Machine Searching Transl., Cleveland, 1959*, pp. 811–822 (1959).

26. Hiż, H., Syntactic completion analysis and theories of grammatical categories, *T.D.A.P.* No. 21, Univ. of Pennsylvania, 1959.

27. Hockett, C. F., The quantification of functional load, RAND Paper 2338, Rand Corp., 1961.

28. Ingerman, P. Z., *A Syntax-Oriented Translator.* Academic Press, New York, 1966.

29. Irons, E. T., and Feurzlig, W., Comments on the implementation of recursive procedures and blocks. *Comm. ACM* **4**, No. 1, 65–69 (1961).

30. Joshi, A. K., A procedure for a transformational decomposition of English sentences, *T.D.A.P.* No. 42, Univ. of Pennsylvania, 1962.

31. Joshi, A. K., String representation of transformations and a decomposition procedure, Part I, *T.D.A.P.*, Univ. of Pennsylvania, Dec. 1965.

32. Joshi, A. K., Transformational analysis by Computer. Paper presented at National Institutes of Health Seminar in Computational Linguistics, Oct. 6–7, 1966.

33. Kuno, S., Automatic syntactic analysis. Paper presented at N. I. H. Seminar on Computational Linguistics, Oct. 6–7, 1966.

34. Kuno, S., and Oettinger, A. G., Multiple-path syntactic analyser, in *Information Processing 1962*, pp. 306–312. North-Holland Publ., Amsterdam, 1963.

35. Kuno, S., The predictive analyzer and a path elimination technique. *Comm. ACM* **8**, 453–462 (1965).

36. Lamb, S. M., On alternation, transformation, realization, and stratification, Monograph Series on Language and Linguistics, No. 17, pp. 105–22. Georgetown Univ., 1964, Inst. of Languages and Linguistics.

37. Lambek, J., The mathematics of sentence structure. *Am. Math. Monthly.* **65,** 154–170 (1958).

38. Lecerf, Y., Une représentation algébrique de la structure des phrases dans diverses langues naturelles. *Compt. Rend.* **252,** No. 2, 232 (1961).

39. Matthews, G. H., Analysis by synthesis of sentences in a natural language. *1st Intern. Conf. Machine Transl. Appl. Language Anal., Teddington, England, 1961* (1961).

40. Oettinger, A. G., *et al.*, Mathematical linguistics and automatic translation, Rep. No. NSF–9, Vol. 2, Cambridge, Mass., 1963.

41. Rhodes, I., Hindsight in predictive syntactic analysis. *Natl. Bur. Std. (U.S.) Rep.* **7034** (1960).

42. Rhodes, I., A new approach to the mechanical syntactic analysis of Russian, Mechanical Translation Group, U.S. Dep. of Commerce, Appl. Math. Div. *Nat. Bur. Std. (U.S.), Rep.* **6595** (1959).

43. Robinson, J., PARSE: A system for automatic parsing of English, RM–4654–PR, RAND Corp., Santa Monica, California, 1965.

44. Robinson, J., Preliminary codes and rules for the automatic parsing of English, RM3339–PR, RAND Corp., Santa Monica, California, Dec. 1962.

45. Sager, N., Morris, J., Salkoff, M., and Raze, C., Report on the string analysis programs, NSF Transformations Project, Dept. of Linguistics, Univ. of Pennsylvania, March 1966.

46. Sager, N., Procedure for left-to-right recognition of sentence structure, *T.D.A.P.* No. 27, Univ. of Pennsylvania, 1960.

47. Salkoff, M., A string grammar of English in Backus normal form, String Project, Inst. for Computer Research in the Humanities, New York University, in preparation.

48. Simmons, R. E., Answering English questions by computer: A survey. *Comm. ACM* **8,** No. 1, 53–70 (1965).

49. Walker, D. E., and Bartlett, J. M., The structure of languages for man and computer: Problems in formalization. Paper presented at the First Congress on the Information Sciences, 1962.

50. Walker, D. E., *et al.*, English Preprocessor Manual, Language processing techniques sub-department, Information Sciences Dept., The Mitre Corp., Bedford, Mass., 1964.

51. Wells, R., Immediate constituents. *Language* **23,** 81–117 (1947).

52. Yngve, V. H., *COMIT Programmer's Reference Manual.* M.I.T. Press, Cambridge, Mass., 1962.

53. Yngve, V. H., A model and a hypothesis for language structure. *Proc. Am. Phil. Soc.* **104,** No. 5, 444 (1960).

54. Yngve, V. H., Random generation of English sentences. *1st Intern. Conf. Machine Transl. Appl. Language Anal., Teddington, England, 1961* (1961).