Cover .... p. 1
Errata .... p. 2
Table of Content .. p. 3
I. A syntactic analyzer for natural language, N. Sager, pp. 4-45
II. The IPL string analysis program, J. Morris, pp. 46-82
III. Outputs from scientific articles, N. Sager & M. Salkoff, pp. 83-143
IV. FAP string analysis program, C. Raze, pp. 144-181

SPR 1

REPORT ON THE STRING ANALYSIS PROGRAMS

Introductory Volume

NSF Transformations Project
Department of Linguistics
University of Pennsylvania
Philadelphia, Penna. 19104
March 1966

REPORT ON THE STRING ANALYSIS PROGRAMS

Introductory Volume

ERRATA

IV FAP String Analysis Program

p. 7, line 12: "to 1-a to set up..." should read "to 1-d to set up..."

p. 23, line 19: "$T_n = S_{il}$ ..." should read "$T_k = S_{il}$ ..."

pp. 33-34: 7 8 11 12 - vertical arrows pointing up should be pointing down.

FIRST REPORT ON THE STRING ANALYSIS PROGRAMS

I. A Syntactic Analyzer for Natural Language - Naomi Sager

II. The IPL String Analysis Program - James Morris

III. Outputs from Scientific Articles - Naomi Sager and Morris Salkoff

IV. The FAP String Analysis Program - Carol Raze

A Syntactic Analyzer for Natural Language

Naomi Sager

Section I describes a procedure for grammatical analysis of sentences of a natural language. The procedure is based upon certain general features of language structure. Section II describes a working computer program for carrying out the analysis of English sentences, using this procedure. The description covers the general plan and the special mechanisms by which the program handles such essential features of language structure as detailed subclass restrictions, coordinate and comparative conjunctions, and syntactic ambiguity.*

I. A procedure for machine recognition of sentence structure**

1. Word-by-word formulation of string structure

The procedure for recognizing the structure of sentences of a natural language is based on linguistic string analysis,*** an axiomatic theory of sentence structure which presents, in terms of particular syntactic categories for words of the language (e.g. N noun, tV tensed verb) a set of elementary

*The program described in this paper was written in IPL V by James Morris and has since been reprogrammed in FAP by Carol Raze. 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.

**Talk presented at the National Science Foundation Seminar on Documentation Research, Washington,D.C., March 6, 1962. This talk summarizes the work presented in the writer's "Procedure for Left-to-Right Recognition of Sentence Structure," Transformations and Discourse Analysis Papers (T.D.A.P.) #27, NSF Transformations Project, Univ. of Pa. 1960.

***Harris, Z. S., Papers on Formal Linguistics, No. 1: "String Analysis of Sentence Structure," Mouton and Co. N.V., The Hague, 1962; a revised edition of "Computable Syntactic Analysis" T.D.A.P. #15, NSF Transformations Project, Univ. of Pa. 1959.

strings of word-categories and rules for combing 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.* 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 useable 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 $\xi = X_1 \ldots X_n$ 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 inter-element point or to the left of X_1 or to the right of X_n, or to the right of an s_ξ which has been adjoined to ξ in one of these manners.

3

*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.

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.*[*]

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

[*] 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 $\underline{\Sigma\ t\ V\ \Omega}$ (*t* = tense). This is the approach adopted in the machine grammar.

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 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 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, every 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
5 by the string-class definitions.

The string relation between successive words of a sentence enables us to establish a simple recognition procedure which goes from left to right through the sentence.* At each successive word-position the procedure compares the classifications (category symbols) of the current sentence word with a list of category symbols obtained by applying the derivation rule at the current point in all the analyses valid to date, i.e. by asking (for each analysis): what is the next symbol of the string currently in progress and what is the first symbol of each string permitted to enter at this point? By the observation above, at least one classification of the current sentence word must match a symbol on the list. The match provides an analysis of the current sentence word and advances by one word-position the particular analysis (or analyses) which contributed the matching string-symbol to the list.

We start out by requiring the occurrence of a center string, because the theory asserts that every sentence has an elementary 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:

$$\begin{array}{rcl} N\ tV\ N & \in & z \\ P\ N & \in & r_N \end{array}$$

Suppose the sentence that we are trying to analyze is

Cars without brakes cause accidents.

N	P	N/tV	N/tV/V	N	.

The first symbol N of the required center string N tV N matches the first

* This observation is made in terms of words, because these are conventionally marked in writing by word-space. In some languages, e.g. the more inflected ones, some of the elements whose succession we would describe in terms of this string relation would have to be morphemes which are not independent words.

sentence symbol N. The sentence word cars is thereby analyzed as the first element of the center string. Now it may be that in a richer grammar this same first symbol, N, might be the beginning of some other string which was permitted at this point. In that case, we would have a second analysis at this point. But in our simple grammar this is not the case.

In the second position we do not get a match of a classification of the sentence word with the current (second) symbol of the center string. So we ask, does this word have a classification which is the first symbol of a string permitted at this point? And, indeed, there is an adjunct class allowed here, namely the right adjuncts of N, and one member of this class (the sole member in this case) is the string P N whose first symbol P matches the classification P of the second sentence word. So we accept this analysis of the second sentence word and insert in our requirement list the string P N (with P already satisfied). We push 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. 7

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 $i^{\underline{th}}$ position means that we have n potential analyses of the sentence which are identical up to the $i^{\underline{th}}$ 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. Indirectly defined strings: *wh-* and *C*.

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 symbol position of the string. While we could establish new axioms for this particular class, and appropriate rules of derivation, it is more 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, consider the sentence *The woman believed the man knew the guest wrote the book*. A number of *wh*-strings can be obtained from this sentence by N-omission. In each case
8 the omitted N can be identified with the N which is adjoined by the *wh*-string, as shown below by the subscripting and the placing of ().

From:

The woman	believed	the man	knew	the guest	wrote	the book.
N_1	tV_1	N_2	tV_2	N_3	tV_3	N_4

We obtain:

the woman	who	()	believed	the man	knew	the guest	wrote	the book.
N_1	wh	()	tV_1	N_2	tV_2	N_3	tV_3	N_4
the man	who	the woman	believed	()	knew	the guest	wrote	the book.
N_2	wh	N_1	tV_1	()	tV_2	N_3	tV_3	N_4
the guest	who	the woman	believed	the man	knew	()	wrote	the book.
N_3	wh	N_1	tV_1	N_2	tV_2	()	tV_3	N_4
the book	which	the woman	believed	the man	knew	the guest	wrote.	()
N_4	wh	N_1	tV_1	N_2	tV_2	N_3	tV_3	()

As the examples illustrate, one can get complicated sentences which, as they get longer, sound less and less acceptable; but one recognizes that they are well-formed sentences of English. Each is obtained by the simple operation of omitting one of the required N's from the inserted wh- string.

As was mentioned, the omission operation may be applied to N in certain adjunct strings. For instance, suppose we have 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 main body of the string and P N_3 (of the book) is an adjunct of N_2 (title).

9

We can form three *wh*-strings which can adjoin N_3: (1) $\underline{wh\ N_1\ tV\ N_2\ P}$ (*the book which the author chose the title of*), omitting N_3; (2) $\underline{P\ wh\ N_1\ tV\ N_2}$ (*the book of which the author chose the title*), omitting $\underline{P\ N_3}$; and (3) $\underline{N_2\ P\ wh\ N_1\ tV}$ (*the book, the title of which the author chose*), omitting N_2 with its adjunct $\underline{P\ N_3}$.

Another class of strings which is best defined by an operation is the class of C-strings headed by coordinate conjunctions *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 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 $\underline{\alpha\ C\ \beta}$ is a C-equivalence and the portion $\underline{C\ \beta}$ is a C-string. For example, let x be the class of center strings, with α the assertion center and β the question. $\underline{\alpha\ C\ \beta}$ gives us such sentences as
10 *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 $\underline{\xi_1\ C\ \xi_2'}$ and $\underline{\xi_1'\ C\ \xi_2}$ 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. E.g. *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 $\underline{\xi_1' \; C \; \xi_2'}$ can also be obtained by the above rule: *to formulate and answer the question*. An operation similar to omission can be defined to obtain complete 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 $\underline{\xi_1 \; C \; \xi_2}$ 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 $\xi = X_1 \ldots X_n$. If a conjunction C appears following X_ℓ in an occurrence of ξ in a sentence $(1 \leq \ell \leq n)$, then the string headed by C which can interrupt ξ at this point is any one of
11 the following:

$$CX_\ell \; ; \quad CX_{\ell-1}X_\ell \; ; \quad CX_{\ell-2}X_{\ell-1}X_\ell \; ; \quad \ldots \; ; \quad CX_1 \ldots X_\ell \; .$$

For example, let ξ = to $V\Omega$ (to repeat the question). In a sentence which begins To repeat the question and answer ..., $\ell = 3$, and corresponding to $C\,X_\ell$ we have to $V\,\Omega\,C\,\Omega$ in which question and answer are both seen as N's. This sequence has another analysis (corresponding to $CX_{\ell-1}X_\ell$) as 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 CX_ℓ where $\ell = 2$ 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:

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

12

3. Restrictions; Ambiguity.

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 sub-categories 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 non-plural verb constitutes a restriction on the N tV N center string. Or we might be able to establish the subcategory: animate nouns. And we could, perhaps, say that certain verbs do not accept, as a noun object, a noun from the subcategory animate. E.g., wear clothes would be acceptable but not wear people.* To illustrate how such a restriction appears in the course of analyzing a sentence, consider the sentence The fur people wear hunters risk their lives for. If we did not have this restriction on the object of wear, after seeing only the first five words of this sentence (The fur people wear hunters), we would have an analysis which treats the fur people as subject and wear hunters as verb-plus-object.

* This is not strictly a grammatical restriction; the sentence She wore a snake around her neck like a necklace is not ungrammatical. The usage restrictions discussed above are useful, however, in practical applications of the procedure in particular fields for which relevant subcategories can be set up.

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 agreement restriction, we could list every string to which it applies, 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 lose the relationship between the strings of each pair.

There are, finally, a few grammatical features which can be fitted into the entire scheme with the aid of restrictions. The article, for instance, can be considered a type of left adjunct which doesn't repeat. It then fits into the machinery of adjunction, but we have to add a restriction that this particular adjunct cannot be repeated.

The machinery for applying restrictions in a recognition program is based on a simple observation that all of the restrictions that we can name operate within a particular scope: either within some string ξ or 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. 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.

Lastly, a few words about 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 grouping symbols into identical formula occurrences by different applications of the rules. For example, in the sentence _People wearing hats is unusual_, the segment *people wearing hats* can be grouped in only one way, as a noun-replacement string agreeing with the singular verb *is*. In the sentence *People wearing hats are unusual* the same segment has a different grouping as a plural noun with a right adjunct 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 groupings 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 are ambiguities permanent and when are they resolved by having the entire analyzed sentence as context? Apparently a permanent ambiguity arises when two sources of ambiguity interact. For example, in *The steel industry requires increases each year*, we have one source of ambiguity in the multiple classification of *increases*. But in order to make a permanent ambiguity we require a second source. The second source in this case is the different groupings of the segment *the steel industry requires*. One grouping considers *industry requires* as an adjunct of *steel*. With this grouping *increases* is a verb. A second grouping sees *steel industry* as a compound noun. With this grouping, *increases* is taken as a noun.

This example illustrates the point that one source of ambiguity alone is not enough to give rise to a permanent ambiguity, and not every pair of sources of ambiguity will give a permanent ambiguity. They have to be such as to interact, as to fit in with each other.

Finally, one might ask: What is the burden of ambiguities in the course of computation? We find that if we proceed in a left-to-right manner as described, some of the 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. A count noun, for instance, without an article, immediately falls down as an element of a center string. It can be a left adjunct of a noun, but it cannot be a center string element without a preceding article (*Book ends are needed*; *The book ends happily*, but not *Book ends happily*). Hence the application of the count-noun restriction, 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).

We also find that some analyses require rare combinations of string occurrences, e.g. an $\underline{N\ V^{-N}}$ string as right adjunct of N where the $\underline{N\ V^{-N}}$ string begins with an extended compound noun. Often the analysis which includes these rare occurrences fails at the end of the sentence. But a practical program could be so arranged as to follow the more likely possibilities first.

Finally, not every analysis need be obtained by applying the procedure. Once one analysis of a sentence has been obtained, certain of its properties may be used to obtain others as departures from the first successful analysis.

II. The IPL Program[*]

1. General description

The analysis routine of the IPL 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 grammar is entered into the machine as a set of definitions, similar in form to B.N.F. definitions. 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 the linguistic restrictions associated with the given string or set of strings. The restrictions are written in a special language of basic

[*] The IPL program is described more fully in James Morris' "The IPL Program for String Analysis" included in a forthcoming set of program manuals and description of the grammar.

routines. Despite the fact that restrictions can be looked upon theoretically as abbreviations for families of unrestricted strings, it would probably be impossible to write a string grammar of a natural language without them, especially if some degree of refinement is desired. In the grammar of English presently used by the program there are approximately 150 strings grouped into about 20 major sets, with about 200 restrictions in all.

(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 proscribed 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 Rl-specify restriction (described below). The definition for _and_, for example, carries an Rl-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.

2. Restrictions

In the IPL 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 construct an instance of the definition in question. 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 several specialized restriction types which are executed before φ is attempted:

R1 (specify) — replaces the given definition of φ with one obtained by executing the R1 test.

R3 (disqualify) — disallows a particular value (option) of φ if such a value would not be well-formed in the given context.

R4 (omission) — accepts the null element of omission or zeroing as value of φ if certain conditions are met.

R5 (SP-scope) — tests whether a special process definition φ should be tried on the next higher level of the analysis tree.

R6 (SP-insert) — tests whether a special node φ (corresponding to a special process definition) should be inserted at the present point in the current string.

R7 (atom-check) disqualifies φ if the current sentence word is not in a word-category which is on the list of atoms 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, an adjunct set which has insertion permission at a given point p in a string φ_j is represented as an element of φ_j at p; however, this element, unlike the "real" elements of the string, either may or may not have a value in the sentence. This convention relieves the program of having to distinguish required, i.e. bona fide, string elements from optional insertions, i.e. adjunct strings, but it adds to the grammar a number of pseudo-strings and elements which represent relations among strings rather than strings of the grammar. Thus, in place of the word-category X (to which left and right adjuncts of X may always be adjoined), there is introduced the pseudo-string $\bar{X} = \underline{\ell_X\ X\ r_X}$, where ℓ_X and r_X are allowed to take on the value zero. And instead of, e.g., the definition

C1 ASSERTION = SUBJECT VERB OBJECT

in which it is understood that sentence adjunct strings may occur at any inter-element position and at the beginning and end of the string, we have

$$\text{C1 ASSERTION} = * \ \text{SUBJECT} \ * \ \text{VERB} \ * \ \text{OBJECT} \ *$$

where * stands for the set of sentence adjunct strings to which the option zero has been added.*

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, the device of transparency was introduced whereby various routines which search the analysis tree "see" only the essential linguistic nodes; the nodes corresponding to pseudo-strings (e.g. $\bar{X}$, above) and psuedo-elements (e.g. *, above) 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 I were to be entered into the machine as the following definitions:

$$z_1 \text{ (center)} = \bar{N}_1 \ \overline{tV} \ \bar{N}_2 \quad \text{[subscript indicates position in string]}$$

$$\bar{X} = \ell_X \ X \ r_X \quad \text{[for X = N, tV, P]}$$

* The actual definition of C1 as used by the grammar (and seen in the output parses) contains the adjunct set r_V (right adjuncts of verb) in the immediate post-object position (for such strings as *with gusto* in *He responded with gusto*). Adjunction at a distance, but always within the scope of a given string, presents no problem for string theory, and is handled with particular ease in the above notation.

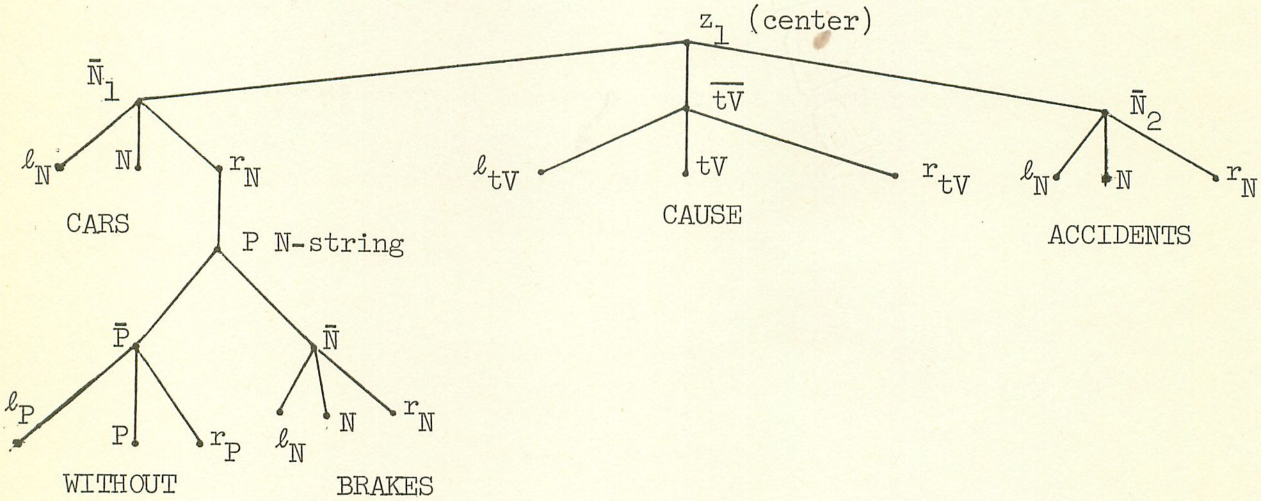
$\ell_N = \ell_{tV} = r_{tV} = \ell_P = r_P = 0$ [i.e. all adjunct sets except r_N are empty]

r_N = - P N-string - 0 [i.e. r_N has as value a P N-string or zero]

P N-string = $\bar{P}\ \bar{N}$

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



Suppose now that the restriction list associated with $\overline{tV}$ contains a wellformedness restriction which checks for number agreement between the verb atom tV (*cause*) and the noun atom of the subject (*cars*). Such a restriction might be: If the word value of tV is definitely plural (singular), then the word value of the subject N must not be definitely singular (plural): $\nexists$ *The car cause accidents*, $\nexists$ *Cars causes accidents*, $\exists$ *The car(s) caused accidents*,

∃ *The series converge(s)*.[*] The restriction would employ the location routines

ascend to X	ascend to the node named X, passing through transparent nodes.
descend to X	descend to the node named X, scanning from right to left all the nodes on each successive level ℓ which lie under transparent nodes of level ℓ-1.

the logical tests

and $X_1 \ldots X_n$	exit + (true) if all the tests $X_1 \ldots X_n$ are true.
imply $X_1 X_2$	exit + if test X_1 is false or if both tests X_1 and X_2 are true.
not X	exit + (-) if test X is false (true).

and the further tests

path: $X_1 \ldots X_n$	exit + if the location routines $X_1 \ldots X_n$ can be executed leading to a value.
attribute X	exit + if the present atomic node has the attribute X.

A test can be included in a path by writing it in parentheses. Letting S0 stand for "definitely plural" and S1 for "definitely singular," the restriction in $\overline{tV}$ would read:

[*] When formulated this way, the test need not be applied in full to verbs which do not distinguish number (e.g. *caused*); also nouns which do not distinguish number (e.g. *sheep*, *series*) need not carry a number attribute. The agreement restriction used by the program is considerably more detailed than the one shown here.

agreement test = and A B

A = imply C D

C = path: descend to tV (attribute S0)

D = path: ascend to z_1, descend to N_1 (not attribute S1)

B = imply E F

E = path: descend to tV (attribute S1)

F = path: ascend to z_1, descend to N_1 (not attribute S1)

Transparency is also used in preparing the output because the tree representation somewhat obscures the fact that a sentence has been decomposed into axiomatic strings. The string character of the analysis is restored in the output display by causing the print routine to ignore transparent nodes. Each non-transparent non-atomic node (i.e. each linguistic string) is assigned a numbered line of output and under each element is printed only the values of non-transparent 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) = $\bar{N}_1$ $\overline{tV}$ $\bar{N}_2$

CARS 2. CAUSE ACCIDENTS

2. P N-string $\bar{P}$ $\bar{N}$

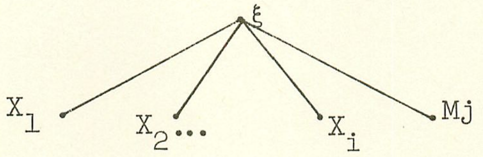
WITHOUT BRAKES

The fact that the number 2. fall to the right of <u>cars</u> reflects the fact that the P N-string on line 2. is a right adjunct of <u>cars</u>.

3. Special Processes

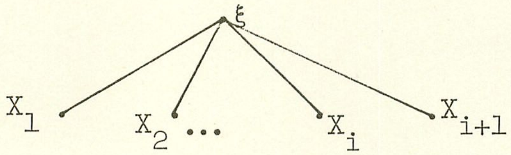
When a word marked "special" (i.e. carrying an M-attribute) becomes current, the program inserts an element Mj (= the value of the M-attribute for the given word) into the current string ξ at a point just following the most recently satisfied element X_i:

FIG. 1



Mj is the name of a grammar definition (M1 for <u>and</u>, M2 for <u>or</u>, etc.) and unless the insertion of Mj is disqualified by an R6-test in Mj, the various options of Mj are then tried. If all the options fail, the program erases Mj and tries X_{i+1} in the normal manner.

FIG. 2



If X_{i+1} is satisfied by the current word (perforce the special word) in one of its non-special capacities (e.g. <u>but</u> as adverb rather than conjunction in <u>They found but one</u>), then the program continues from this point as usual. The program will automatically succeed if X_{i+1} is an adjunct set because of the presence of a zero option in an adjunct set appearing as a string element; and if the remaining elements of the current string are all adjunct sets the

the program will reach the end of the current string without using up the special word. It will then ascend (unless blocked by an R5-test) and repeat 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 non-special) fail, the program backs up to the previous sentence-word, as it would normally on failure. This scheme is especially apt for conjunctions.

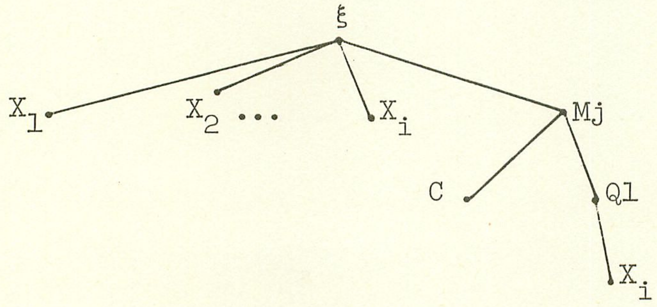
Coordinate conjunctions: *and*, *or*, *but*

The Mj-definitions for coordinate conjunctions all contain an option whose content is specified by an R1-restriction as follows: Given that Mj has been inserted following X_i in ξ (Fig. 1), the restriction produces a set

$$Q1 \;=\; -X_i \;-X_{i-1}X_i \;-X_{i-2}X_{i-1}X_i \;- \;\ldots\; -X_1 \ldots X_{i-1}X_i \;,$$

i.e. carries out the 'structural repetition' of Section I. Suppose that in the situation pictured in Fig. 1, the first option X_i of Q1 succeeds; the analysis tree would appear:

FIG. 3



E.g. for ξ = *to* $V\ \Omega$, this could correspond to *to* $V\ \Omega\ C\ \Omega$ (*to repeat the question and answer*; *answer* taken as a noun). We say that Q1 is a "shadow" ξ; the restrictions housed in ξ are also applied at Q1, care having been taken in the writing of the restriction so that it works correctly on truncated strings. Taken together, the method of inserting Mj-nodes plus the specification of Q1 enable the program to produce all possible conjunctional strings at the point in the analysis where the conjunction occurs, i.e. all C-strings which might occur (in some sentence) following the portion of the sentence analyzed to date.

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 enables us to extend a number of restrictions so that they cover conjunctional strings (which contain the same restrictions as non-conjunctional strings). For example, the sentence *The present paper describes quantitative analyses and other studies also* might have an analysis as $\underline{\Sigma\ tV\ \Omega\ C\ \Sigma}$ (similar to *He was here and she also*) except for the disagreement in number between the subject and (zeroed) verb of the conjunctional string: $\nexists$ *... and other studies describes quantitative analyses*. Another example is the usage restriction which requires that an appropriate subclass of N-subject occur with a given verb. E.g., the word *studies*, qua verb, can be said not to occur with a chemical noun as subject, so that in such a sentence as *This protein was used in various analyses and studies*, the non-wellformed *The protein was used ... and the protein studies* can be ruled out. In view of its usefulness for restrictions and possibly for the extension of the present work toward text-analysis, zeroed elements are supplied for certain C-strings, in particular those based on the assertion string. An example is seen in the output for sentence 6 (Fig. 4) from a protein chemistry test.

FIGURE 4

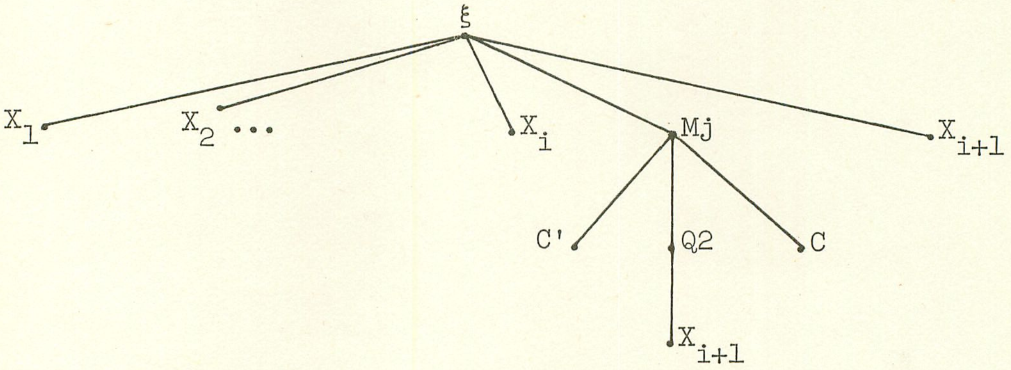
```
THE SENTENCE. THE PRESENT PAPER DESCRIBES THE QUANTITATIVE AMINØ ACID ANALYSIS AND ØTHER
FUNDAMENTAL STUDIES ØF PRIMARY IMPØRTANCE TØ THE SEQUENCE DETERMINATIØN .

     SENTENCE        = INTRØDUCER  CENTER   MI        END-MARK
                                   3        AND 10    .

  3. C1 ASSERTIØN    =  *  SUBJECT      *  VERB          *  ØBJECT       R-V     *
                           1 PAPER         DESCRIBES        2 ANALYSIS

 10. Q1 CØNJUNCT     = CENTER
                       9

  1. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                       THE                  PRESENT

  2. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE                  TYPE NS  NØUN
                       THE                  QUANTITATIVE AMINØ                  ACID

  9. C1 ASSERTIØN    =  *  SUBJECT                   *  VERB             *  ØBJECT          R-V    *
                           ( THE PRESENT PAPER )        ( DESCRIBES )       4 STUDIES 8

  4. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE               TYPE NS  NØUN
                                            ØTHER FUNDAMENTAL

  8. C20 P N         = L-P  P    N
                            ØF   5 IMPØRTANCE 7

  5. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                            PRIMARY

  7. C20 P N         = L-P  P    N
                            TØ   6 DETERMINATIØN

  6. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                       THE                                      SEQUENCE
```

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, of the two sentences This is a theory which work on price supports (i.e. work on price supports a theory) and This is a theory which works on price support, the first is not ambiguous by virtue of the lack of agreement in number between theory and work (which would be required if theory were subject of work). To apply the agreement restriction usefully in this case, it is necessary for the program to "know" that the null value of the subject in the wh-string actually stands for the word value of a particular element (theory) in the host string, and to carry out the subject-check on this word. In general, when a restriction path arrives at an element whose value is the null of an omission or a zeroing operation, the path is rerouted automatically by means of an "omission list," a particular list of location routines which leads to the adjoined element (omission) or the structurally parallel element (zeroing) of the host string.

Scope-marked conjunctions: Either ... or, neither ... nor, both ... and.

In terms of the operation of structural repetition by which conjunctional strings are obtained, the scope-marker (C') 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. $\exists$ *John wrote a paper and Mary corrected it*, $\exists$ *John both wrote and corrected a paper*, $\nexists$ *John both wrote and Mary corrected a paper*. I.e., a scope-marker-and-conjunction pair *C' ... C* 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, where X (Q2 in the grammar) is specified to be the currently expected string or string-element (extendable to string-segment); when this string is satisfied, the program reverts to its normal operation and finds X as it expected before the interruption. E.g.

FIG. 5



The tree in Fig. 5 could correspond to *to C' V C V Ω* (*to both formulate and answer the question*). Here C is a non-special element, because it is not an independent string head, but part of *C' X C*.

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) while in other uses it is punctuational and as such hardly subject to grammatical description. Furthermore, as conjunction, while it is most often associated with other conjunctions (as in X, X and X) so that the absence of a forthcoming conjunction can be used to disqualify this particular option, it also occurs as a "that is"-type conjunction which has no such convenient marker (It strikes at the root of a scientific school if one accuses it of insufficient knowledge of facts, of insufficiency of basis of induction). However, this last type is relatively rare. (The program is equipped with a "rare" switch, such that strings marked "rare" are not used unless no other analysis is possible.) It is possible, therefore, to try the comma first as a conjunction (Q1 option with various restrictions), and if no conjunctional analysis is possible, to accept the comma as punctuation. The words following the comma must then be analyzed by the ordinary grammar; in addition, certain strings which normally occur set off by punctuation commas carry their own restrictions to this effect, e.g. The children asleep, they ...

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*
Are you for or against the plan?	*Are you more for than against the plan?*
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.*
Men buy books and women flowers.	*More men buy books than women flowers.*
	Men more often buy books than women flowers.
	∃? *Men buy more books than women flowers.*

With a few restrictions on the placing of the comparative marker in the host string, the large body of comparative strings can be obtained by the same process (Q1) as the conjunctional strings.

(2) *wh*-like:

There exists in the grammar a "sentence adjunct *which*-string," i.e. a *wh*-string with omission which adjoins a whole sentence, e.g. *The crowd dispersed quietly, which had not been expected.* The ultimate verb in this string must be of the type which can have a sentence (in the form *that S*) as its object (e.g. *expect*), or of the type which can have *that S* as subject (e.g. *surprise*: *That the crowd dispersed quietly surprised many*). For the former case, there is a similar comparative string:

The crowd dispersed quietly, which had not been expected.

The crowd dispersed more quietly than had been expected.

The crowd more quickly dispersed than had been expected.

More of the crowd dispersed quietly than had been expected.

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

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 theordinary 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 Ql 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: $\Sigma\ tV\ \Omega\ C\ t\ \Sigma$

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) Semi-idiomatic, for particular words such as *ever*:

> *more beautiful than ever*
>
> *more restrictive than necessary*
>
> *more restrictive than usual*

4. The treatment of syntactic ambiguity

Saving

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 ($m < n$) 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 IPL V string analysis program therefore is equipped with a device called "saving" by which computed instances of certain strings (substructures of the analysis-tree) are copied before being dismantled in the course of the program's backing up to obtain an alternative analysis of the words in the substructure. These substructures are then available for re-use as "plug-in" units in other analyses or paths.

The question immediately arises: Is it "safe" to use in a given context a structure which was constructed in a different context? For example, a *P N* or *N P N* string in the scope of a *wh*-string (or other "omitting" string) can be well-formed while lacking the N of *P N* ; the same *P N* (actually consisting only of P) would not be well-formed as a part of a different, non-omitting string. E.g., in the ambiguous sentence

They called the man to give the job to John,

one analysis contains the N-omitting string *to V* Ω^{-N} (*to give the job to*) as a right adjunct of N (*man*). In this analysis the *N P N* object of *give* actually consists of *N P* (*the job to*) due to the omission of N, as is required somewhere in the scope of *to V* Ω^{-N}. (In this analysis the object of *call* is *N N*: *They called the man John.*) A second analysis contains as right adjunct of *man* the non-omitting string *to V* Ω, i.e. *They called the man* (*who is*) *to give the job to John*. In computing this analysis it would be incorrect to use the previously obtained *N P* (*the job to*) as the *N P N* object of *give*, since the context is no longer an omitting string. (A third analysis of this sentence would take the same *to V* Ω string as a sentence adjunct, i.e. as from *in order for them to give the job to John*.)

A structure which contains no restriction that looks outside that structure and doesn't examine context is called _autonomous_, and can without question 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. It behooves the grammar-writer to make as many strings as possible autonomous. However, it should be noted that this aim conflicts with the desire to house restrictions as close to the atomic level as possible, i.e. as low in the tree structure as possible, so that non-wellformed sequences will be discarded soon after their construction.

However, even non-autonomous structures can be saved if the process of constructing that structure with the given words has not had recourse to restrictions or parts of restrictions which examine the outside context. As a practical measure, then, many restrictions are in the form of an implication in which the if-clause checks for some feature which separates the autonomous from the non-autonomous cases to make use of such differences and to make it possible to save a particular instance φ of a non-autonomous structure, when this is justified, the program keeps track of the paths followed in executing restrictions during the construction of φ; and φ is saved only if no restriction path extended beyond φ (to a parent or sibling node on the tree).

Freezing

Given one analysis of an ambiguous sentence one can divide the alternative analyses into those which involve a different breakdown of the sentence into axiomatic strings (strings of the grammar), and those which retain the same strings but derive their presence from a different use of the rules for inserting strings into other strings, i.e. affect only the external relations of the given strings. An example of the former was seen in the ambiguous sentence *The steel industry requires increases each year*. The shifting of *P N* strings from one adjunction class to another (e.g. r_N to r_V), or from adjoining one particular noun in the sentence to adjoining another (e.g. $\underline{P\ N_3}$ as adjunct of N_1 or N_2 in $\underline{N_1\ P\ N_2\ P\ N_3}$), is a rich source of ambiguities of the latter type, as in *He broke the window with the wooden frame* and *I know a man with a car from France*.

Among the alternative analyses which contain a string (or strings) not found in the first analysis, one can further distinguish those which involve a different segmentation of the sentence into stretches which correspond to strings, from those which merely re-analyze one of the given string-stretches into a case of a different string. An example of the latter is the sentence *The men work without a break until given hours for rest* in which *until given hours...* is either a *P N* (like *until particular hours*) or a passive subordinate-conjunctional string *Cs Ven Ωp*: *until (they were) given hours for rest*.

"Freezing" has to do with the possibility of resegmenting differently. If we can establish that the words of an already-computed structure φ have no other possible segmentation in the given sentence, then it would be foolish to break up φ in order to seek other analyses of the same words. Rather, φ should be treated as a unit and manipulated by other analyses as a unit, as though it were a single word or a frozen block. Thus, where it has been possible to state conditions under which the words comprising an instance of a structure cannot be resegmented differently, these have been incorporated into a "freeze-test" in the definition of the structure. 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 substructure) from the analysis-tree as a unit. It should be noted that this does not imply that on proceeding forward again and encountering the first word of φ the program will assume that φ should be re-attached; in the changed context the program may be asking for quite another structure. (Thus, ambiguities of the type illustrated just above are not lost by applying the freeze-test. In the example, *until given hours...* might be frozen as a *P N*, but this would not prevent the program in its forward phase from asking in the usual way for other structures permitted at the given point in the analysis -- here, asking for *Cs Ven Ωp* beginning with *until*.) Nor is it even assumed that if the program asks for the same structure again at this point, then φ will be re-attached as a unit, since an instance of a non-autonomous structure may be freezable without being savable.

As an example, consider the distinguished segment $\underline{\ell_N\ N}$ (N with its left adjuncts). If the sentence-word satisfying N (the "core value" of $\underline{\ell_N\ N}$) is a count noun, then except under statable conditions the stretch reaching from a preceding article (or other word capable of satisfying the count-noun restriction) through the count noun cannot be differently segmented. 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 $\underline{\ell_N\ N}$, can be "frozen." The statable exceptions mentioned above involve re-analyzing the N as a left adjunct of N, e.g. as part of a compound noun or compound adjective. As the ambiguity of the following examples shows, the underlined stretches are instances of $\underline{\ell_N\ N}$ where N is a count noun but $\underline{\ell_N\ N}$ cannot be frozen:

The organic origin traces reveal reasons against the hypothesis.

The spray free corners attract flies to them.

Who wants *a microbe* breeding disease?

They wanted *the slate* elected officers.

The freeze-test for $\underline{\ell_N\ N}$ says then: $\underline{\ell_N\ N}$ can be frozen if both the following are true: (1) one analysis of the sentence has already been obtained (this clause is included in all freeze-tests); (2) the core value of $\underline{\ell_N\ N}$ is a count noun which is not followed in the sentence by a word with one of the classifications N (noun), A (adjective), Ving, or Ven (past participle).

Min-word test

By far the largest number of alternative analyses are the result of associative ambiguity: The breakdown of the sentence into axiomatic strings in the various analyses is the same but the grouping is different (including shifting a string from adjoining a particular word class to adjoining the string as a whole). 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*). If the first adjunct itself ends in N, then, unless there are further constraints to apply (e.g. subclass affinities), an ambiguity results, since the second adjunct can adjoin either the original N (repeatability) or the final N of the first adjunct (nesting). The most common situation is a sequence of *P N*'s in r_N: *N P N P N ...* (*the house of my friend in Philadelphia* ...). However, the same type of ambiguity arises with subordinate conjunction strings and other sentence-adjuncts which can themselves have final sentence adjuncts: *I went there even though I was delayed because I had many things to do*.

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*.

r_V: *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. We should also be able to recognize certain P N's as sentence-adjuncts, such as those stating experimental conditions: *at 100°*, *for two hours*, etc. But additional work on subclassifications is necessary to establish these restrictions.

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 and nesting, the output prints the analysis due to nesting. Where strings in r_N, r_V and s.a. overlap, the output assigns the common string to the first adjunct set having permission to enter in the given situation; e.g. in the examples of *P N* in r_N, r_V, and s.a., above, unless we provide restrictions which exclude *with interest* and *with time to spare* as right adjuncts of *box*, the program would print only the analysis of each sentence in which the *P N* is r_N, because r_N is the first of the three sets to have adjunction-permission after an object N (*He opened the box with the blue cover with interest*, ∄ *He opened the box with interest with the blue cover*). This suppression of alternative analyses which are due to associative ambiguity is accomplished with the "min-word test."*

The basis of the min-word test rests 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 m^{th} 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 Cx beginning with word m will only result in an associative ambiguity if any previous analysis contains Cx as an adjunct string beginning

* It should be emphasized that the suppression of associative 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.

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 α_1, α_2 ... etc., which we do not wish to have the program shift from one adjunct set 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). Each α_i as member of an adjunct set carries the following R3-disqualify restriction: 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.

Report on the String Analysis Programs

II

The IPL String Analysis Program

James Morris

I. Introduction

This paper describes a program for natural language parsing. The approach has been to develop a fairly general, language-independent processor and then provide it with the definitions for a language with which it is to parse input sentences. The program is written in IPL-V and the grammar definitions and input sentences are represented as IPL list structures.

II. Representation

A. Grammar

The grammar definitions are the usual set of productions found in any formal language specification.*

sentence = - sub. verb obj. period

verb = - tensed verb - aux. + untensed verb

etc.

The symbol " - " is used to separate the various alternatively possible <u>strings</u> which might be particular manifestations of the grammatical entity defined by the production.

B. Words

A word is represented by a list of its parts of speech. Specifically,

(study) = noun, tensed verb (plural), untensed verb

might be the dictionary entry for the word "study."

* [The examples of grammar definitions used throughout this part of the Report are not the ones used by the actual program.]

Figure 1

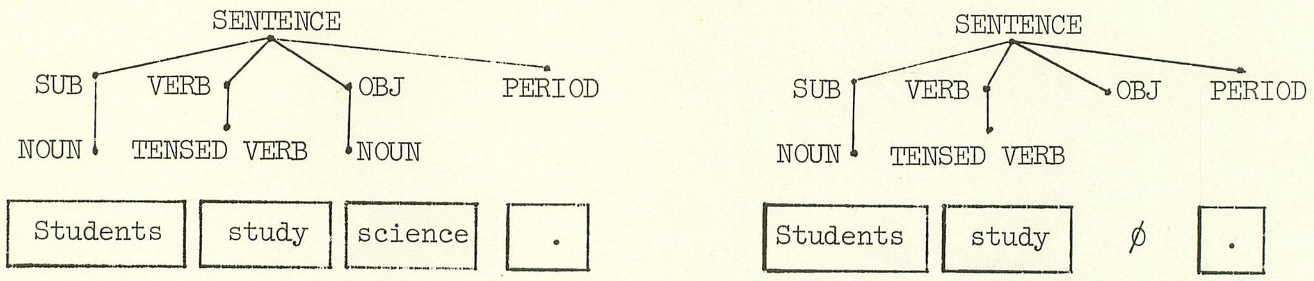
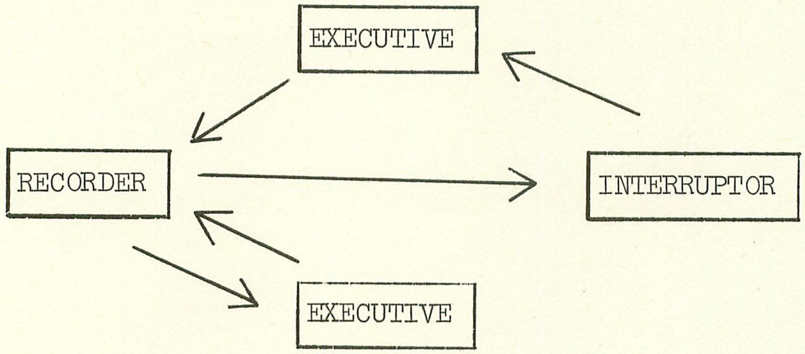


Figure 2



C. The Parse Tree

The output of the program is a tree structure representing the parse of the input sentence in terms of the defined grammar. Each node of the tree corresponds to some grammatical entity and its sub-nodes correspond to the entities named on one of its strings. Figure 1 shows two trees corresponding to parses of "Students study science" and "Students study." The definitions of obj and sub are

Sub = - noun - pronoun

Obj = - noun - pronoun - ∅

where ∅, the null symbol, means that obj might be empty.

III. The processor.

The processor is very simple in its basic form. It merely generates all possible sentences as defined by the grammar, guided by the input sentence; i.e., as soon as it becomes "apparent" that a certain sub-structure is not to be found in the particular input sentence, the generation process by-passes that particular tack and tries another sub-structure. The following is a brief example of how the processor would parse "Students study." The process might be viewed as a question-answer dialogue between the processor and itself (the process is recursive).

```
Is "Students study" a sentence?
      Does "students" begin a sub?
            Does "students begin a noun?   YES
                  YES ("Students" begins a sub)
```

```
Does "study" begin a verb?

     Does "study" begin a tensed verb?   YES

               YES ("study" begins a verb)

Does "." begin an obj?

     Does "." begin a noun?    NO

     Does "." begin a pronoun?   NO

     Does "." begin ∅?    YES  (automatically)

               YES ("." begins an obj)

Does "." begin a period?   YES

     YES ("Students study" is a sentence)
```

Figure 2 shows the general flow of control in the processor. The executive handles the general tree construction and incrementing of the sentence position. The recorder saves copies of certain common sub-structures which the executive has constructed but has since taken off the tree. The interrupter recognizes special words and makes appropriate insertions in the grammar definitions to accommodate the words (such as "and," "or," "than"). Roughly speaking, the executive decides what grammatical entity or sub-structure should be generated and passes control to the recorder which checks in its records to see if the desired sub-structure has already been generated and saved. Usually the answer is "no" in which case the recorder calls on the executive to solve its own problem. If the executive successfully generates or modifies a sub-structure (or fails to do so) it reports its success (or failure) back to the recorder which may save the information. Whatever the case, the recorder passes control to the interruptor which examines the current sentence word for special attributes. The normal case is for the interruptor to find none and promptly pass control back to the executive with the information as to the success or failure at the lower level. Thus the normal flow of control is between the executive and itself; the recorder may short circuit the downward question chain by retrieving the answer from past history, or the interruptor may delay the ascent of answers by demanding further processing at the lower level.

The Executive

The executive routine is given the name of a node in the tree and a current position in the sentence and is expected to either construct a subtree on that node (if no tree yet hangs from it) or modify the subtree (if one has already been constructed). The node has been tagged with the grammatical entity whose structure is to be followed. Thus the user of the program is in effect the super-executive that gives the executive a node, tagged with the entity which is the definition of a sentence, and the initial word of the input string he wishes to be parsed. The entity with which the node is tagged will be called the node value.

The decision process is as follows:

1. Is this a new node? (i.e., with no tree hanging from it)

   If not, go to line 15

2. Is the node value atomic? (having no definition, such as "noun")

If not, go to line 5

3. The node value is atomic. Does it match the current sentence word? (Is "Students" possibly a noun?)

   If not, exit to the interruptor, signalling -, meaning no parse for this entity and this word.

4. The word matches the atomic entity

   1) make the next word in the sentence the current word.

   2) exit to the interruptor, signaling +, meaning the entity given is a parse for the current word.

5. The node value is not atomic. Therefore a subtree must be generated from this node value. Take the first string in the node value's definition and make it the current string.

6. Take the first entity named on the current string and make it the current element.

7. Construct a node, tag it with the current element, and pass control to the recorder.

8. Did the recorder signal +, meaning a subtree for the current node-with-value has been formed, or -, meaning no subtree has been constructed.

   If -, go to 10

9. The recorder signalled +, the current element has been successfully matched to the input string. If there is a next element on the current string make it the new current element and go to line 7. Otherwise, the string is complete; and thus has generated a structure for the node value, exit

+ to the recorder.

10. The recorder signalled -, the current element did not give rise to a subtree. Erase the current node.

11. Is there an element preceding the current element in the current string?

    If no, go to 13.

12. If yes, perhaps modifying the subtree previously constructed for that element to be satisfied. Therefore, backup by making that preceding element the current element, its corresponding node the current node. Pass control to the recorder and go to line 8 to await the results of the subtree modification.

13. There is no preceding element; i.e., the current element is the first element of the current string. Therefore this string cannot be satisfied. Does the node value have a next string in its definition?

    If it does, make that string the current string and go to line 6.

14. There is no next string. Therefore this entity (the node value) cannot be satisfied. Exit - to the recorder, signaling no parse.

15. This node is not new. Is the node value atomic? i.e., are we at the bottom of the tree? If yes, consider the original match of this atomic entity to the sentence word spurious. Make the preceding sentence word the current word, and exit - to the recorder.

16. The node value is not atomic. Therefore there is a subtree hanging from

this node. Make the string and element which were current when this node was finished current again, pass control to the recorder and go to line 8 to await the results of the subtree modification.

The Recorder

The function of the recorder is purely auxiliary to the procedure. Its only purpose is to cut down processing time. Certain entities may be declared savable by the user. When the value of one of the nodes passed to the recorder is savable, the recorder will recreate, if it can, the entire subtree which would normally be generated; or, if no subtree could normally be generated, the recorder will exit to the interruptor signalling -. The reason the recorder is able to do this is that the executive always exits to the recorder with the result of a generation attempt. The recorder simply files the information about savable entities in the following way: The initial subtree generated for an entity (or the signal that there is no initial subtree) is associated with the word that was current when the generation attempt began. The i-th subtree generated for an entity (grammatical ambiguity permits several parses for the same entity) is associated with the (i-1)-th subtree.

The following is a more explicit description of the recorder's actions upon receiving control from the executive:

1. Is the current element a savable entity?

If not, go to line 8

2. The entity is savable. Is the associated node a new one?

   If not, go to line 5

3. The node is new. Therefore an initial parse is desired. Does the current word have a file associated with this entity?

   If not, go to line 8

4. Get the initial record in the file and go to line 6

5. The associated node is not new. Does the subtree hanging from it contain a "next record" indicator?

   If not, go to line 8

   If so, get the next record.

6. Does the record signal "no parse"?

   If yes, pass control to the interruptor, signalling -.

7. The record is a subtree. Attach this subtree to the current sub-node and pass control to the interruptor, signalling +.

8. Call on the executive to construct a subtree.

   If the entity is not savable, pass control to the interruptor, signalling + if the executive signalled +, - if the executive signalled -.

9. The entity is savable:

   -- If a file for this entity at this word has already been started, attach the "result" from the executive to the last record in the file.

   -- If no file has been started, start one, make its initial record the "result" from the executive, and associate the file with the current word.

("result" is the subtree if the executive signalled +, and a "no parse" signal if the executive signalled -.)

10. Pass control to the interruptor, signalling + (-) if the executive signalled + (-).

The Interruptor

The purpose of the interruptor is to allow the user an alternative method of specifying grammatical rules associated with special words. The prime example is the word "and"; it may be used to link almost any two grammatical entities which have similar structures. We would rather not double the number of definitions necessary to specify a language by defining, for every entity X in the original grammar, a new entity X',

$$X' \;=\; - X - X \quad \text{"and"} \quad X \, .$$

The following system is used to handle such cases:

1. The user specifies any words he wishes to be special, and associates with each word a "special" grammatical entity. This entity has a definition the same in form as any other entity (except for a tag marking it special, and possibly a special test; tests are described later).

2. The interruptor, when it receives control from the recorder or the executive, checks the current word to see if it is special. If it is, the interruptor may, depending on various conditions, take the word's associated

special entity, make it the current element, and call on the executive to construct a subtree for that element. When the called executive exits back to the interruptor, the interruptor will in turn exit to the calling executive, signalling + or -, depending upon various conditions.

The interruptor's actions are as follows:

1. Is the current word special?

    If not, go to line 9

2. Is the special word a special prefix type or a special suffix type?

    If prefix, and the recorder signalled -, go to line 6

    If suffix, and the recorder signalled +, go to line 3

    Otherwise, pass control up to the executive, signalling + (-) if the recorder signalled + (-)

3. Suffix-type: Remove the special tag from the word and insert the special entity associated with the word after the current element in the current string. Make the special entity the current element of the string, construct a node, tag it with the current element and call the executive.

4. If the executive exits +, pass control up to the calling executive, signalling +.

5. If the executive exits -, delete the special entity from the current string, make the element preceding it the current element, restore the special tag to the current word, and pass control up to the calling executive, signalling +.

6. Prefix type: Remove the special tag from the word. Insert the associated special entity before the current element in the current string, and make the special entity the current element. Create a new node, tag it with the current element and call on the executive to generate a subtree.

7. If the executive signals -, restore the special tag to the word, delete the special entity from the current string, make the next element on the string the current element, and pass control up to the executive, signalling -.

8. If the executive signals +, pass control up to the executive, signalling +.

9. Is the current element to the string a special entity?

   If not, pass control up to the executive, signalling +(-) if the recorder signalled +(-).

10. Did the recorder signal +?

    If yes, pass control up to the executive, signalling +.

11. Is the special entity a prefix type or a suffix type?

    If prefix, go to line 7

    If suffix, go to line 5

Further Remarks on the Interruptor

On the surface, the only action of the interruptor is to insert special entities into certain instances of string definitions, triggered by the appearance of a so-called special word. The power of this mechanism is realized only when the special entities contain definitions which construct their own strings at run-time, via the mechanism of specification which will be described shortly. For example, the special entity for the special word "and" might contain the machinery to retrieve the subtree which had just been completed before the "and" appeared and to construct from this subtree a definition which would give rise to an isomorphic structure which, when placed after the "and," would satisfy some rule of parallelism.

Interrupt points within the executive (Restrictions)

Aside from the rather heavy-handed interruption by the interruptor, the path of the executive may be altered by the action of several other tests which are built into the grammar (the interruptor, remember, is called into action by information attached to words). These tests are essentially restrictions and extensions of the basic grammar as defined by a set of productions. For instance, the definition of "sentence" appearing above might carry the restriction that the subject noun (i.e. the noun which is the subject) and the verb must be of the same class regarding number -- this in alternative to defining two types of sentences, one for singular subjects and verbs and one for plural.

The following sections describe how these augmentations to the grammar are effected.

Restriction lists

In any grammar definition the "-" appearing at the beginning of a string may be replaced by the name of a restriction list. This list must be a list of pairs; the first member of a pair is an indicator of what type of test or action is to be taken and when, and the second member of the pair is a list of specific parameters for the particular action. For example the definition

sentence adjunct = - pn - ad.phr.

might be altered to be

sentence adjunct = .1 pn .2 ad. phr.
.1 = R1 .10 R2 .21 ,
.2 = R3 .5,

where symbols beginning with "." are understood to be names of lists (not grammatical entities).

There are four principal identifiers which may appear on this list: R1 (specification), R2 (wellformedness restriction), R3 (disqualification), R4 (omission disqualification).

- R1, dynamic specification of definitions.

When R1 appears on a restriction list of the first (N.B.) string in a definition the following action is taken by the executive:

When the executive is at the point of choosing the first string to be tried for a particular entity (Line 5 of executive description), it checks the restriction list of that first string for an R1 indicator. If one appears, the list named immediately after the R1 is retrieved. This list must also be a list of pairs. The first member must be the name of a location list and the second member must be some alphanumeric information to be used when the program is being traced. Location lists are described below; suffice it to say here that a location list describes a path to be traced through the existing structure of the analysis tree (and perhaps into the input sentence). The executive takes the first location list on the R1-list and attempts to trace the path it describes. If the path can be successfully followed (following this path will sometimes be called "executing the location list"), it will lead to the name of a grammatical entity. The executive then uses this found grammatical entity as the effective node value; i.e., the strings in the actual node value's definition will not be tried; rather, the strings in the definition of the found entity will be used. (Usually the location lists used in specifications end up going to some word in the input sentence (such as the word holding a verb position in the analysis tree structure) and thence to some grammatical entity associated with that word (such as the possible objects of that word when used as a verb).) If the path cannot be followed, the next location list on the R1 list is tried, and so on. If no location list can be completely executed, the node value is allowed to stand; i.e. as if R1 had not appeared at all.

- R2, Wellformedness Restrictions

When the executive has successfully completed the generation of a structure for a string (Line 9) it checks the string's restriction list for an R2. If none is found it proceeds as usual and exits + to the interruptor, but if one is found it retrieves the list named after the R2 and does the following:

The list must be a list of pairs; the first member of each pair is the name of a test and the second is a piece of alpha-numeric information (as in an R1 list). Tests are described below. A test is a list which asks some question about the present structure of the analysis tree and signals + (for test satisfied) or - (test not satisfied) to the executive. If any test on the R2 list signals -, the executive does not proceed to exit to the interruptor but backs up to get an alternative structure for the string just completed (goes to Line 16). If all tests are satisfied the executive proceeds to exit as usual.

- R3, Disqualification

When the executive is about to start the generation of a new string (Line 6) it checks that string's restriction list for the appearance of an R3. If an R3 appears, the name of a list must follow it. This list must have the same format as an R2 list -- test, alpha-word, test, alpha-word,... If one of the tests is not satisfied the executive does not bother with the generation of this new string but looks for the next string on the effective node value's definition (goes to Line 13). Otherwise the executive proceeds as usual with the current string.

- R4, Omission disqualification

Immediately after the check is made for regular disqualification (R3), the executive checks the current string's restriction list for an appearance of R4 (assuming all R3-tests were satisfied). If R4 is found, the list named after the appearance of R4 must have the same format as an R1 list -- location list, alpha-word, location list, alpha-word,... If one of the location lists can be executed successfully, it is expected to lead to the name of a node in the tree. To this node is then attached a tag (*7) and the name of a list containing the name of the node value of the current node. If the tag already appears at the found node, the current node value is simply added to the associated list. Further, a tag is attached to the current node (R4) and the name of the found node is associated with the tag. If no location list can be executed successfully the executive skips the current string as when all the R3 tests appear.

The R4 mechanism is used in only one special context, omission. The string bearing an R4 on its restriction list should be a one-element string whose only element is a null entity (an atomic entity which is satisfied automatically, without comparison to the sentence word). For instance,

```
subj. = - noun - pronoun  .1  A61 ,
   .1 = R4  .2
   .2 = L1  (om)  L2  (om) ,
```

would in effect say that the subject could be omitted if either of the location lists, L1 or L2, could be successfully executed. The conditions under which something like a subject might be omitted usually involve the sentence in which the subject appears being a part of a wh-string such as a who-clause. Thus the location list will "climb" up the tree from the point where the subject occurs as a node value looking for a node value which is a wh-string entity. If none is found, omission (letting -A61 be tried as a string of subj.) will not be permitted. If the proper node is found however, the information that an omission has been made will be left at that node in the form of the *7 tag. The reason that the R4 mechanism leaves this information as to its action is to allow other mechanisms to test whether omission has occurred in various places (by looking for a *7 tag). This is useful when one wishes to express a rule such as "one and only one omission must occur within this string."

Tests

A test list as mentioned above is a list whose first symbol is the name of a test and whose remaining symbols are parameters of the form expected by the particular test. The following tests are available to the user:

/0 (no parameters)

Always exits + (true).

/10 X1 ... Xn

And. Exits + if each of the tests X1 ... Xn exits +.

X1 ... Xn must be the names of test lists.

/7 A1 ... An

Exits + if the node value of the present (current) node is any one of the grammatical entities A1 ... An.

/8 (no parameters)

Exits + if the present node has no node below it with a non-null atomic entity for a node value; i.e., if the structure below has been generated without consuming any sentence words.

/9 A1 ... An

Exits + if any node above the present node has node value among A1 ... An.

/11 X1 ... Xn

Or. Exits + if any one of the tests X1 ... Xn exit +.

/12 X1 X2

Implies. Exits + if X1 exits - or if X1 and X2 exit +.

/13 Y1 Y2

Equals. Y1 and Y2 may be

1) any symbol beginning with a character other than "." -- in which case the symbol stands for itself.

2) a symbol beginning with "." -- in which case the following symbols constitute a location list (list of location routines) and is to be executed. If the location list exits + (successful) the found symbol is understood to be the value of Y1 or Y2. If the location list exits -, /13 exits -.

If the value of Y1 is equal to the value of Y2 /13 exits +.

/14 Y1 Y2

Contains. Y1 may be, as above, a symbol standing for itself or the name of a location list. Y2 may be a symbol standing for itself, the name of a location list, or, in addition, the name of a list whose first symbol is 0 (zero).

If either Y1 or Y2 is the name of a location list, the location list is executed. If a location list so executed exits -, /14 exits -. If Y1 or Y2 is not a location list, it is assumed to be its own value. The value of Y2 is expected to be a list. If the value of Y1 is contained on this list, /14 exits -.

/15 X1 ... Xn

Exists. The symbols X1 ... Xn must form a location list (see below). This location list is executed and if its execution yields a + signal, /15 discards the found symbol and exits +, meaning that the path described exists.

/16 X1

Not. The symbol X1 must be the name of a test list. If the test exits +, /16 exits - ; if the test exits -, /16 exits +.

/17 Y1 Y2

Intersection. Y1 and Y2 must be of the same form as the parameter Y2 for a /14. The values of Y1 and Y2 are expected to be lists. If these lists have a common element (other than J3 or J4, which are often used as place-holders on lists), /17 exits +.

Location Lists

Location lists may be thought of as small programs whose effect is to change the contents of a single communication cell which we shall call *0. *0 contains the symbol or node which the location procedure is "looking at." This symbol or node will be called the present node or symbol to distinguish it from the current node which the executive is supposed to be "looking at." This present node is initially the current node; i.e., the path each location list describes starts from the current node. Remember that every location list is ultimately associated with a certain grammatical entity via some restriction list for one of that entity's strings. The current node, then, is a node that has that grammatical entity as its node value.

A certain routine (P17) which we shall call the locator interprets these location lists. Describing its procedure is the simplest way to describe the format and meaning of location lists. In the following description, the current symbol being interpreted by the locator will be referred to as Yi and i will be initially set to 1, making Y1 the first symbol on the location list.

The Locator

1. Is Yi one of /20, /21, ..., /49 ?

If not, go to 2.

If so, execute the symbol (/20 ... /49 are subroutines and are described below).

If the "/" routine exits -, exit -.

If the "/" routine exits +, increment i by the particular routine's number of parameters +1 and go to 6 (The routine's parameters, if it has any, appear immediately after it on the list and must be skipped over.)

2. Is Yi the name of a list (begin with ".")?

    If not, go to 4

    If so, is the first symbol on list Yi a test name (/0 ... /19)?

      If not, go to 3

      If so, execute the test list. If the test exits +, set i to i + 1 and go to 6 If the test exits -, exit -.

3. Yi is the name of a location list within the main location list. Execute Yi. If the execution yields -, exit -. Otherwise set i to i + 1 and go to 6

4. Does the symbol Yi begin with the character "X"?

    If not, go to 5 Otherwise, assume Yi to be a temporary storage cell containing the name of a node. Put that name into *0 (thus setting the present node to some new value; see /45 which puts nodes into these X cells). Set i to i + 1 and go to 6

5. The symbol Yi is not recognizable. Assume it is the name of a tag (sometimes called node attribute). Search the present node for the tag Yi. If not found, exit -; otherwise place the symbol associated with that tag (the "value" of the node attribute) in *0, set i to i + 1 and go to 6

6. Is there an i-th symbol on the location list? If there is, go to 1.

    Otherwise, the location list has been successfully completed. Exit +, and return the final contents of *0 to the calling routine.

"/" routines.

/20 (no parameters)

Ascend one. Replace the present node by the one immediately above it in the tree. If there is no such node, exit -.

/21 X

/21 is also an ascend command, but the length of the ascent is unspecified. Rather, the ascent stops when a node with a particular node value is reached. If no such node is found, /21 exits -. More specifically:

- The parameter X must be a list of the form Y Z1 ... Zn where the Zi's are name of grammatical entities, and Y is a list of grammatical entities.

- There is understood to exist a permanent list of entities called transparent entities. This list is specified by the user.

- The procedure is to ascend (apply /20) repeatedly until one of the following occurs:

  1. The node value of the present node is on the list Y. Then exit + (The desired node has been reached.).
  2. The node value of the present node is not on the list Y, among the entities Z1 ... Zn, or on the list of transparent entities. Then exit -. (The node cannot be reached.)
  3. /20 exits -, then exit -. (The top of the tree has been reached.)

Figuratively speaking, /21 looks up the tree; it can see through only those nodes whose values are transparent, or one of the Zi. If it can see a node whose value occurs on the list Y, it ascends to it. Otherwise it exits -. (The actual program allows the user to be more casual about the format of the parameter structure; i.e. substituting a single entity for a list or a list for a group of entities.)

/22 X

The parameter X is expected to be a list of grammatical entities. /22 ascends repeatedly until a node with a value on list X is encountered. If none is encountered, it exits -. Thus /22 is simply an unsophisticated /21.

/23 (no parameters)

Go left. /23 goes to the immediate left sibling of the present node if one exists. Otherwise it exits -.

/24 (no parameters)

Go right. /24 goes to the immediate right sibling of the present node if one exists. Rather than exit -, if one doesn't exist, as will often be the case in a partly constructed tree, /24 goes to the string definition which will generate the next node, and looks for an element after the current element. If one exists, it in effect constructs a temporary node whose value is that element and makes it the present node. If no next element exists /24 exits -. Thus a location list can effectively test yet-to-be constructed nodes.

/25 - not in use

/26 - not in use

/27 (no parameters)

/27 is a special purpose routine to detect and follow "parallel" paths in a tree structure. Starting from a node in one sub-structure, /27 ascends to a node whose value is Q1 (the special entity for a conjunction), ascends 2 more nodes, and then descends along a path described by the node values encountered on the way up from the original node.

More specifically, /27's instructions are:

1. Execute /20 (ascend one). If /20 exits -, exit -.
2. If the present node value is Q1, got to _4_
3. Put the present node value at the top of push-down list, L.
   Go to _1_
4. Ascend twice.
5. Set i = 2
6. If there is no i-th symbol on list L, exit +.
7. Is there a node immediately below the present node whose value is the i-th symbol on list L?

   If not, exit -.
8. Otherwise make that node the present node, set i = i + 1, and go to _6_

/28 X

/28 is necessary only to overcome a special feature of the notation which allows the user to define an entity locally without naming it. For example, B23 = -(- A60 - A13) A14

defines the entity B23 to be a two-element string the first element of which is an entity (nameless) which may be either one of the strings - A60 or - A13. In order to go from a node whose value is B23 to one whose value is the locally defined entity (- A60 - A13), the command "/28 A13" or the command "/28 A60) is given.

/30 (no parameters)

Descend one. /30 looks for a node below the present one. If there is none or more than one, it exits -. Otherwise the present node becomes that single node.

/31 X

/31 is the descending analogue of /21. The nodes below the present node are examined under the constraint that non-transparent nodes may not be passed through. The parameter X must have the same form as the parameter of /21:

X = Y Z1 ... Zn

Y = W1 ... Wn ,

where the Wi are entities being searched for in the subtree and the Zi are the non-transparent entities that may be passed through. The specific decision procedure is as follows:

1. Initialize a push-down list L, to be empty.
2. Set node A to be the present node.
3. Scan each of the nodes below node A. If one of them has node value

among the Wi, make that node the present node, and exit +. If more than one has node value among the Wi, choose the left-most (or last constructed) one. Otherwise place each of the nodes whose value is either transparent or among the Zi at the end of list L.

4. If list L is empty, exit -. Otherwise, pop-up list L and make the new node at the beginning of list L node A. Go to line 3

By examining this procedure one can see that /31 will scan all the nodes that can be reached by descending in the tree without going through non-transparent nodes. Further, the search will look at the nodes closest to the beginning node first.

/32 X

/32 is the descending analogue of /22. It scans the nodes below it in the same manner as /31 but ignores the constraint of transparency. Therefore /32 will search every node below the present node. If a node whose value is on list X is found, that node is made the present node and /32 exits +.

/33 through /39 not in use.

/40 X

Alternate path operator. The parameter X is a list whose elements are names of location lists. /40 executes each of these location lists, starting from the present node. If they all exit -, /40 exits -. Otherwise the present node becomes the node reached by the first location list whose execution yields a +.

/41 X (Y)

/41 has two modes:

1. If the symbol X is a "/" routine, then Y is assumed to be the parameter of that "/" routine if a parameter is required. The routine X is executed repeatedly until it exits -. If it exited minus after the first execution, /41 exits -. Otherwise (if there was at least one successful execution of X), /41 exits +.

2. If X is not a "/" routine it is assumed to be a two-member list, and Y is not considered as a parameter of /41. The first member of list X is the name of a location list; the second member is a test. The action is as follows:

   1) Execute the test. If it exits +, exit +. Otherwise,

   2) Execute the location list. If it exits -, exit -. If it exits +, go to 1

   Thus, in node 2, /41 is analogous to an iteration statement or "Do loop."

/42 X

Retrieve and execute location list X. The parameter X is assumed to be a location list whose value is the name of another location list. The action of /42 is to use the location list X to retrieve a pre-stored location list, and to execute that location list starting from the present node. If either location list cannot be executed (exits -), /42 exits -.

/43 X

The symbol X is, in fact, assumed to be a symbol whose first character is "X." /43 save the name of the present node in the cell X and exits +. The present node does not change.

/44 (no parameters)

/44 is a general purpose routine. It takes the contents of the present node and makes it the present node. Thus the meaning of /44 varies with the type of cell which is the present node. Some instances are:

present node before	operation	present node after
node in tree	/44	node value of that node
node in tree	*1 /44	list representing sentence word
node in tree	*3 /44	name of restriction list of current string of current node value

/45 X

Quote. The symbol X is made the present node.

/46 (no parameters)

/46 is a very special-purpose routine used for R1 specification of special entities associated with co-ordinating conjunctions. It constructs a grammatical entity from information in the tree in the following way:

1. Define the present node value to be Xi.
2. Define the node value of the immediate left sibling of Xi to be X(i-1), and so on down to X1.

3. Construct the definition Y.

$$Y = - X(i-1) - X(i-2)\ X(i-1) - \ldots - X1 \ldots X(i-1)$$

4. If there is no immediate left sibling of the present node, exit -. Otherwise leave the name Y in *0 (making Y the present node. Note that Y is not a node in the tree but the name of a grammar definition.).

/47 X

/47 assumes that the present node is not really a node but the name of a grammar definition, as might be left by /46. X is the name of a test list. /47 edits the definition using the test X as the criterion for a string being allowed to remain in the definition. Since the present node is not a node in the tree, the location lists used by the test X may not meaningfully use many of the "/" routines. In fact, the only elementary ones that will make sense are /24, /44, and /45. Because of the particular machine representation chosen for grammar definitions the following are examples of acceptable forms for X.

X = /15 /24 (/7 A1 A2 A3) will delete all strings from the definition whose first element is not A1 or A2 or A3

X = /14 (/24 /24 /44) (0 A5) will delete all strings whose second element is not A5.

X = /16 (/15 /24 /24 /24) will delete all strings with three or more elements.

X = /41 ((/24)(/7 B2)) will delete all strings which do not have B2 as some element.

Additional Aspects of the Program

Implicit Disqualification

There are really two uses for the application of a disqualification test for a particular string. One is to disallow the generation of a string in a certain context where it would be successfully constructed but the user does not wish sentences with such a construction to be allowed. The second use is to save computing time. If a certain entity has a very complex structure whose attempted generation will take a significant time, even if the final result is a failure, the user might wish to attach some quick feasibility tests in the form of disqualification to one or more of the entity's strings. The first test one might think of for these purposes is a comparison of the current word with all those atoms which might begin the string. Since this type of test might be used very often, rather than have the user write an R3 test every time he wishes to have such a test made, we introduce a new possible attribute, R7, of the restriction list. When an R7 appears on a restriction list of a string the symbol following it must be the name of a list of atoms (the atoms which can begin the string). Whenever the executive is about to begin the generation of a new string it looks for the R7 attribute on that string's restriction list. If it is found and the current word has no atomic value coinciding with one on the R7-list, that string is passed over and the next string sought. Thus an R7 disqualification has the same effect as an R3 disqualification. The user may attach an R7-list himself when defining the entity, or have a program generate the R7 lists for that entity.

∅-location lists

Consider the following situation: There is a restriction attached to all verb-strings which says in effect, "After this string is completed, descend to the core word in this verb-string and note its specific atomic value. Then ascend to the sentence-type string that contains this verb-string, descend to the subject, descend to the core value of the subject, and compare the subject's core value with the verb's for number agreement." Note that if this process cannot be carried out to completion (i.e., if the path describing the location list cannot be successfully executed), the entire restriction will fail. Thus it behooves the person writing location lists to make sure that the location list does, in fact, always reach some node, when the intent of the location list is not simply to check for the existence of a path.

For, suppose the input sentence were "The boy who eat the apple...," which should not be recognized as a legal sentence by any reasonable program. The definition for a "who" clause might be:

who-clause = "who" sent

where, in order to satisfy the sentence string in the who-clause the program has had to choose "- A61" (null of omission) as the value of subj. in that sentence string. When, after "eat" has been successfully analyzed as the verb-string of the sentence, the agreement restriction is applied, it will retrieve the atom corresponding to "eat" and start off to get the core atom of the subject. Depending on how "descend to the core value of the subject" is specifically written in the location list, we will either signal -, meaning no core value exists, or present A61 as the core value. In either case, the restriction is not going to serve the purpose of checking subject-verb number agreement because the atom that should be compared to the verb atom in this case is boy, which is in the subject subtree of the main sentence. One solution to this problem is simply to "let the user beware"; that is, in writing location lists he must test every node value that might be null-by-omission and, when one is, take an alternative path provided in the location list which would lead to the node to which the restriction should be applied. The solution used by the program is rather to introduce ∅-location lists (∅- for ∅mission) which allow the user to specify this path once and for all for each omission. When a location routine lands on an A61 it is automatically rerouted to a new location via the ∅-location list.

For example, we might redefine subj. to be

```
subj.  =  - noun - pronoun .1 A61 - noun who-clause,
   .1  =  R4  .2  ∅  .3,
   .2  =  L1 (om) L2 (om)
   .3  =  "a location list"
```

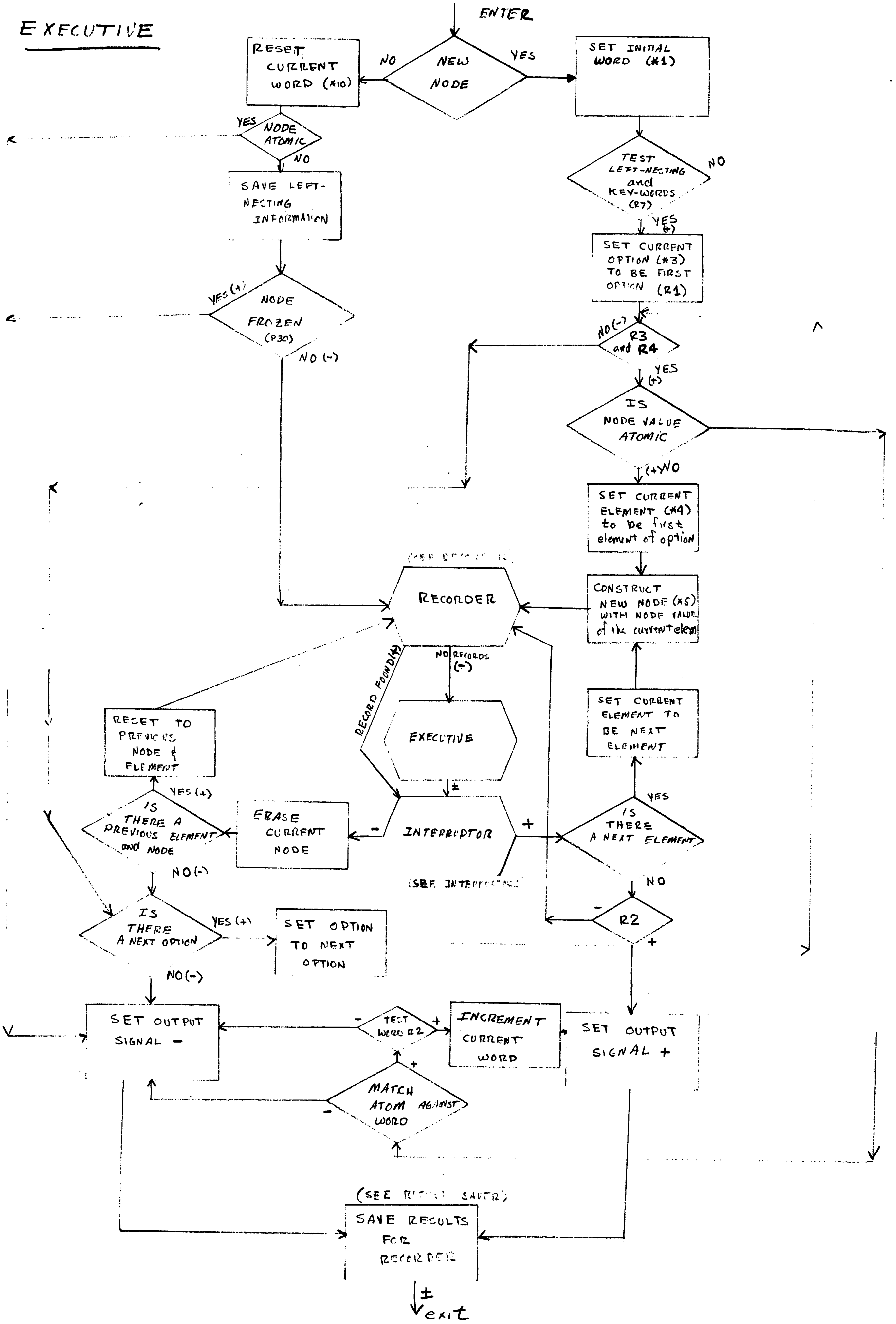
The effect of doing this would be as follows: Every "/" routine that descends (primarily /30, /31, /32) will, when it cannot find what it's looking for by the usual means, check the nodes it encounters for strings that have ∅-lists. Finding one, it will follow the path described by that ∅-list; and, if it can do so successfully, it will continue the downward search from that node located by the list.

The ∅-list for the example above might say, in effect, "Go up to the sentence node (/21 (sent.)); Go up to the who-clause node (/21 (who-clause)); Go up to the node with a value "like" the omission node's value (/21 (subj.)).

Freezing

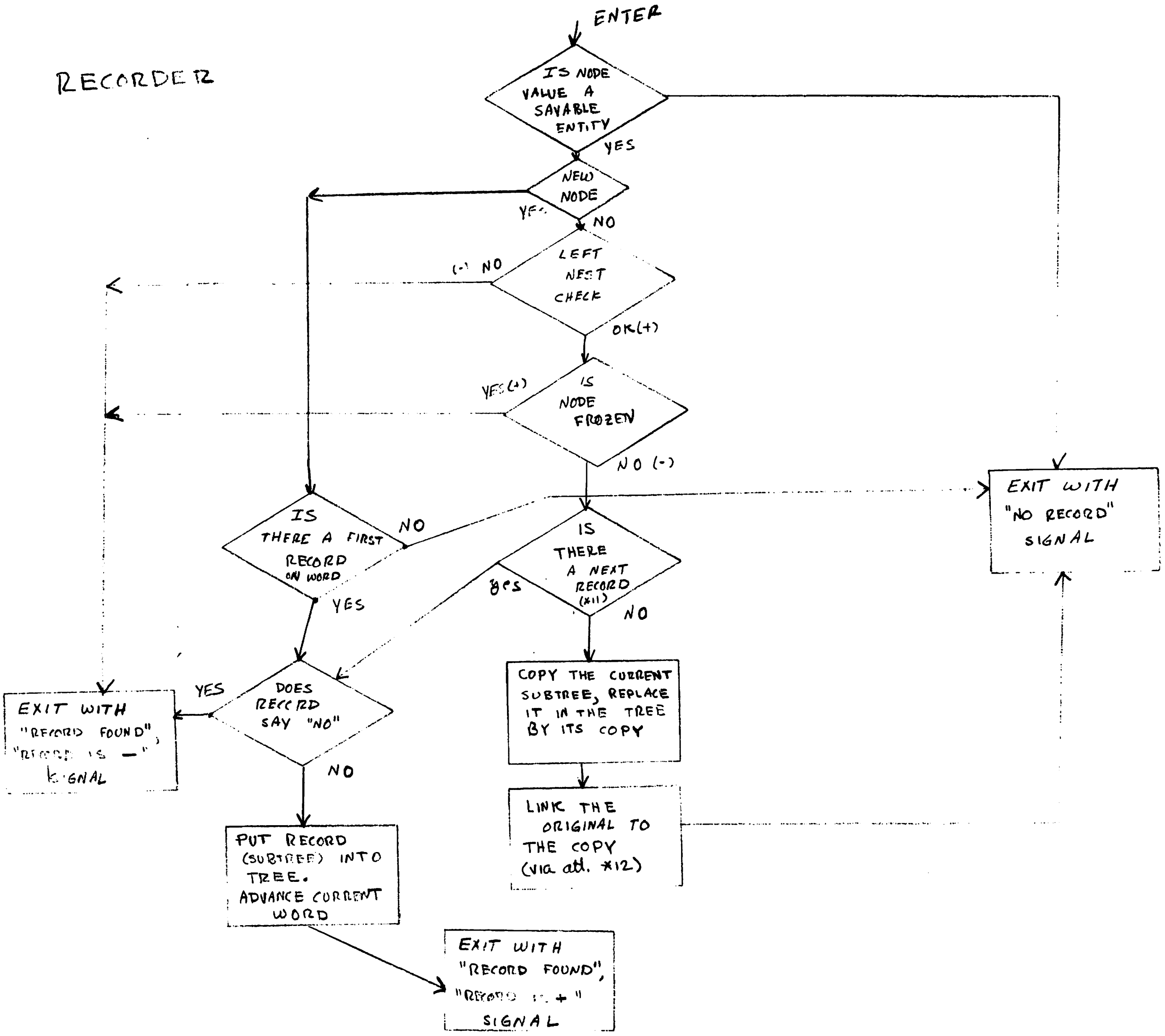
Under certain conditions the user may want to say, "Once the program has generated one subtree for this particular entity it shouldn't try backing up to get alternate analyses because there are no ambiguities in this particular structure. To effect such an action the user places the attribute F on the restriction list of the first string of the entity he wishes to freeze. The symbol following the symbol following the attribute F must be the name of a test list. When the executive is descending to a node that is not new (line 16), it checks the entity for a freeze test and finding one executes it. If the test exits -, the executive proceeds as usual; but if the test exits +, the executive immediately exits - to the interruptor. Thus the effect is as if the executive did, in fact, try to get an alternative parse but failed.

EXECUTIVE

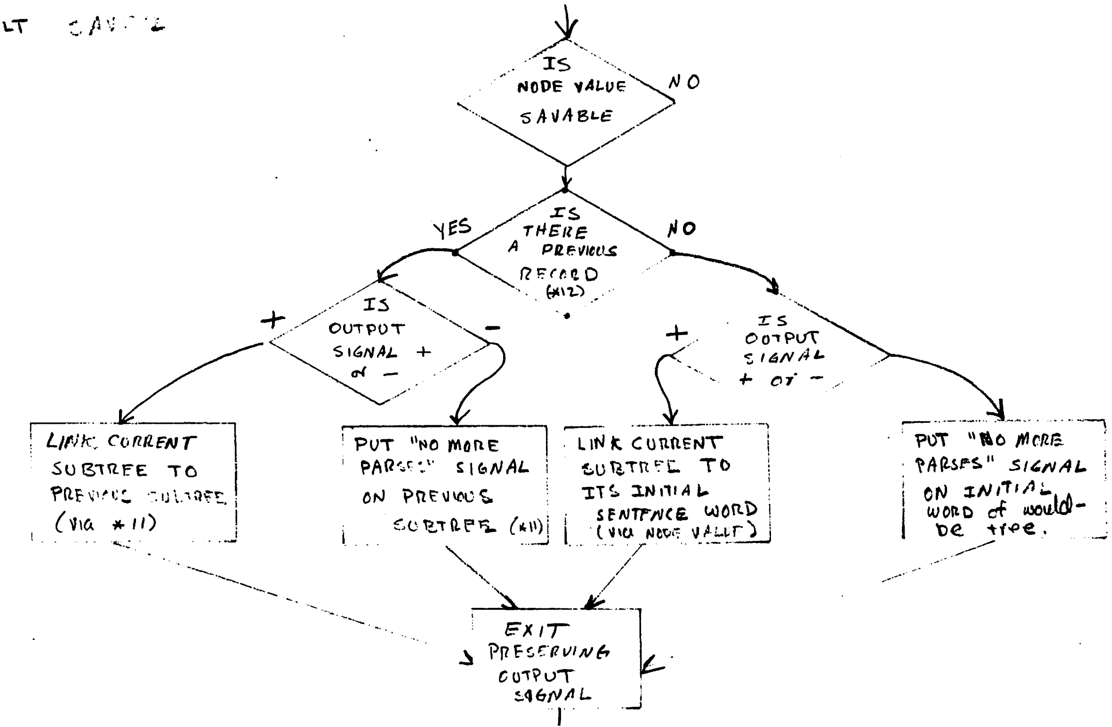


ENTER
NEW NODE
NO
YES
RESET CURRENT WORD (*10)
SET INITIAL WORD (*1)
YES
NODE ATOMIC
NO
TEST LEFT-NESTING and KEY-WORDS (R7)
NO
SAVE LEFT-NESTING INFORMATION
YES (+)
SET CURRENT OPTION (*3) TO BE FIRST OPTION (R1)
YES (+)
NODE FROZEN (P30)
NO (-)
NO (-)
R3 and R4
YES (+)
IS NODE VALUE ATOMIC
(+) NO
SET CURRENT ELEMENT (*4) to be first element of option
(SEE RECORDER)
RECORDER
CONSTRUCT NEW NODE (*5) WITH NODE VALUE of the current elem
NO RECORDS (-)
RECORD FOUND (+)
SET CURRENT ELEMENT TO BE NEXT ELEMENT
RESET TO PREVIOUS NODE & ELEMENT
EXECUTIVE
±
YES (+)
YES
IS THERE A PREVIOUS ELEMENT and NODE
ERASE CURRENT NODE
-
INTERRUPTOR
+
IS THERE A NEXT ELEMENT
(SEE INTERRUPTOR)
NO (-)
NO
-
R2
IS THERE A NEXT OPTION
YES (+)
SET OPTION TO NEXT OPTION
+
NO (-)
SET OUTPUT SIGNAL -
-
TEST WORD R2
+
INCREMENT CURRENT WORD
SET OUTPUT SIGNAL +
+
MATCH ATOM AGAINST WORD
-
(SEE RESULT SAVER)
SAVE RESULTS FOR RECORDER
±
exit

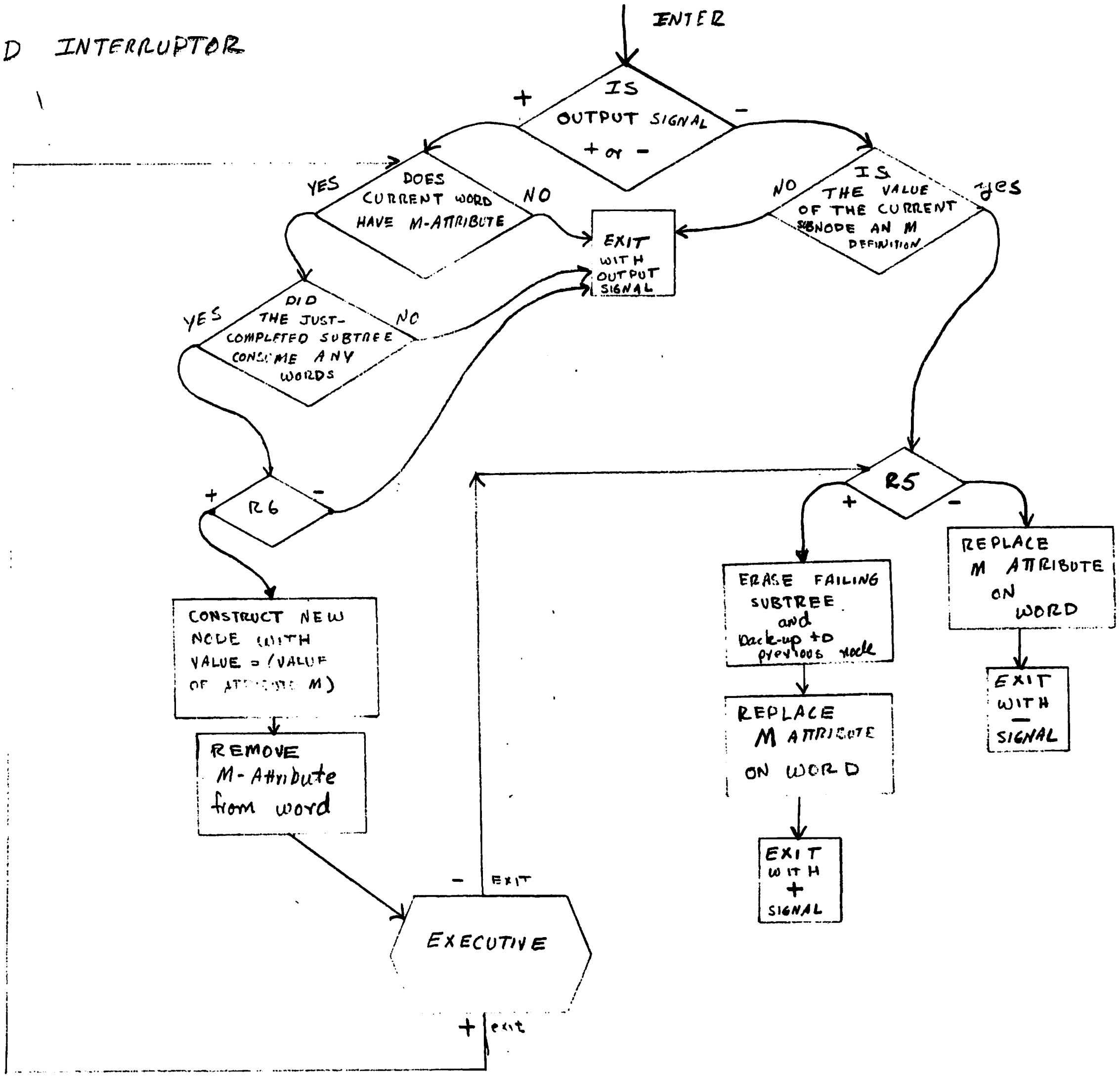
RECORDER


ENTER
IS NODE VALUE A SAVABLE ENTITY
YES
NEW NODE
YES
NO
LEFT NEST CHECK
(-) NO
OK(+)
IS NODE FROZEN
YES(+)
NO (-)
IS THERE A FIRST RECORD ON WORD
NO
YES
IS THERE A NEXT RECORD (*11)
yes
NO
EXIT WITH "NO RECORD" SIGNAL
DOES RECORD SAY "NO"
YES
NO
EXIT WITH "RECORD FOUND", "RECORD IS —" SIGNAL
PUT RECORD (SUBTREE) INTO TREE. ADVANCE CURRENT WORD
COPY THE CURRENT SUBTREE, REPLACE IT IN THE TREE BY ITS COPY
LINK THE ORIGINAL TO THE COPY (via att. *12)
EXIT WITH "RECORD FOUND", "RECORD IS +" SIGNAL

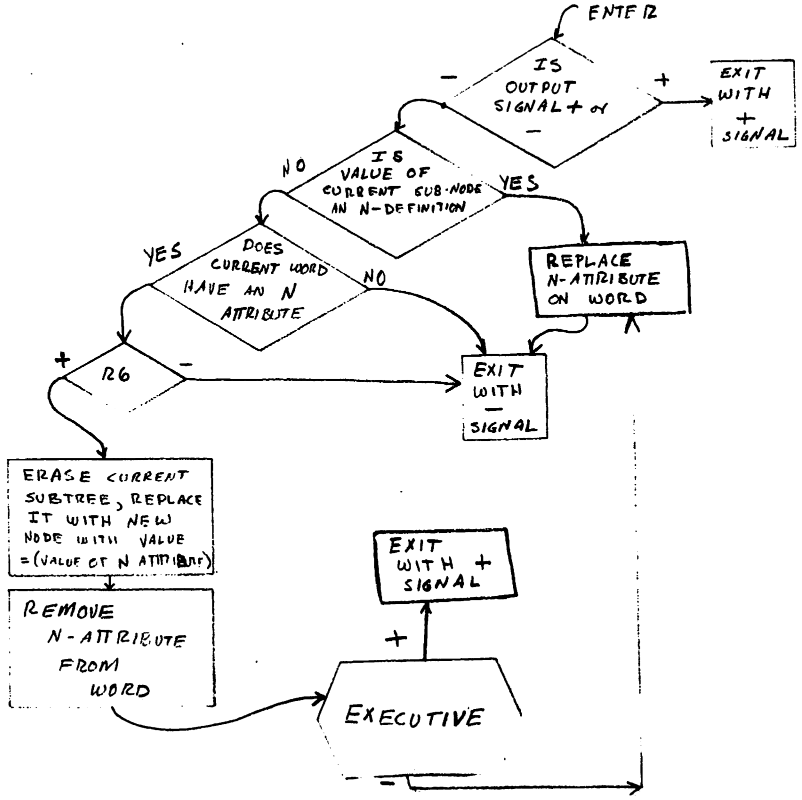
RESULT SAVER


IS NODE VALUE SAVABLE
NO
IS THERE A PREVIOUS RECORD (*12)
YES
NO
IS OUTPUT SIGNAL + or −
+
−
IS OUTPUT SIGNAL + or −
+
LINK CURRENT SUBTREE TO PREVIOUS SUBTREE (VIA *11)
PUT "NO MORE PARSES" SIGNAL ON PREVIOUS SUBTREE (*11)
LINK CURRENT SUBTREE TO ITS INITIAL SENTENCE WORD (VIA NODE VALUE)
PUT "NO MORE PARSES" SIGNAL ON INITIAL WORD of would-be tree.
EXIT PRESERVING OUTPUT SIGNAL

FORWARD INTERRUPTOR


ENTER
IS OUTPUT SIGNAL + or -
+
-
DOES CURRENT WORD HAVE M-ATTRIBUTE
YES
NO
EXIT WITH OUTPUT SIGNAL
IS THE VALUE OF THE CURRENT SUBNODE AN M DEFINITION
NO
yes
DID THE JUST-COMPLETED SUBTREE CONSUME ANY WORDS
YES
NO
R6
+
-
R5
+
-
CONSTRUCT NEW NODE WITH VALUE = (VALUE OF ATTRIBUTE M)
REMOVE M-Attribute from word
ERASE FAILING SUBTREE and back-up to previous node
REPLACE M ATTRIBUTE ON WORD
REPLACE M ATTRIBUTE ON WORD
EXIT WITH - SIGNAL
EXIT WITH + SIGNAL
EXECUTIVE
- EXIT
+ exit

BACKWARD INTERRUPTOR


ENTER
IS OUTPUT SIGNAL + or -
-
+
EXIT WITH + SIGNAL
IS VALUE OF CURRENT SUB-NODE AN N-DEFINITION
NO
YES
REPLACE N-ATTRIBUTE ON WORD
DOES CURRENT WORD HAVE AN N ATTRIBUTE
YES
NO
R6
+
-
EXIT WITH - SIGNAL
ERASE CURRENT SUBTREE, REPLACE IT WITH NEW NODE WITH VALUE =(VALUE OF N ATTRIBUTE)
REMOVE N-ATTRIBUTE FROM WORD
EXIT WITH + SIGNAL
+
EXECUTIVE
-

REPORT ON THE STRING ANALYSIS PROGRAMS

III

Outputs from Scientific Articles

Naomi Sager and Morris Salkoff

Each successive sentence of a scientific article is entered into the computer without any pre-editing. The program looks up each word of the sentence in a "dictionary" which gives for each word all its grammatical classifications without reference to the way the word is used in the given article. Thus, the program does not use the meanings of words or any special adjustment to the particular article being analyzed. The program then decomposes each sentence into a very short elementary sentence which is the grammatical center of the original, plus various strings of words: each string has a fixed grammatical structure, and adjoins the elementary sentence or one of its adjoined strings.

Form of the output

Each string is assigned a numbered line of output on which is written the name of the string, and, to the right of the equal sign, the successive elements of the string. The successive elements of a string are in most cases (atomic) word classes: noun, verb, adjective, etc. In certain cases an element of a string is a named sequence of word classes, called here a string-segment. On the output line the name of the element is written above and the word (or words) in the sentence which is the value of that element is written below it. If the value of an element is a string-segment whose internal structure is of interest, the string-segment is written on a separate line and the number of the line is written as the value of the element in the original string. For example, in the output line[*]

```
10.  C1 ASSERTION  =  * SUBJECT      * VERB     * OBJECT R-V  *
                        1 PURITY 3     IS         4  9
```

from the output for sentence 8A (*The high purity of the sample is also demonstrated by the analyses reported below*), the first element (subject) has the value *purity*, the second element (verb) has the value *is*, and the third element (object) has as value the string segment written on line 9.

According to string theory, sentences are built up by the insertion of strings into other strings at stated points: (1) To the left or right of an atomic element X there can be inserted, respectively, any string in the set ℓ_X (left adjuncts of X) or r_X (right adjuncts of X). E.g., for X = N noun, we may find to the left of N such left-adjunct-strings of N as an article, adjective, etc., and to the right of N such right-adjunct-strings as a prepositional phrase P N, a *wh*-string, etc. Each left and right-adjunct string (except the ordered left adjuncts of N which are treated as a sequence) is written on a separate output line and the number of the line is written to the left or right, respectively, of the word to which it is adjoined. For example, in output line 10, above, the number 1 to the left of *purity* indicates that *purity* in this sentence has the left-adjunct(s) written on line 1 of the output, and the number 3 to the right of *purity* indicates that *purity* here has the right-adjunct written on output line 3. (2) At various points in particular strings, one may insert sentence-adjunct-strings. The set of sentence-adjuncts includes interjections (*however*, *moreover*, *in general*),

[*] The computer examples used here are from the glucagon text outputs.

subordinate conjunction strings (*because he went home*, *when at sea*), certain P N, time expressions (*last week*), and others. The points at which these strings can be inserted are indicated by asterisks. If a sentence-adjunct-string occurs in the sentence at a given point *, the string is written on a separate line and the number of the line is written under the * in the original string. For example, in output line 10, above, the number 4 indicates that a sentence-adjunct, written on line 4, occurred between the verb and object of C1.

Several features of the output require additional explanation, particularly the treatment of the several types of null elements of the grammar. Particular verbs can occur with a zero-object (*He slept*, *Glass breaks*). In these instances the object element has nothing written below it. An example of this occurs in sentence 12 (*The hydrolyzates were processed in a manner similar to that described by Smith and Stockell*) where the passive object of *described* in output line 6 (below) is zero:

```
7.  C20  P N       =  L-P  P    N
                           TO   3  6

3.  L-N            =  ARTICLE  QUANTIFIER  ADJECTIVE  TYPE N'S NOUN
                      THAT

6.  C132 PASSIVE   =  VEN          *  PASSIVE-∅   R-V    *
                      DESCRIBED 5
```

Another type of null element is a zero noun, which occurs when certain sequences of words occupying a noun position can be analyzed as a particular sequence of left-adjuncts of N not followed by a noun, e.g. *the poor*, *the impassive*, *the following*, *his mother's*, *five*, *several*, *this*, *that*, etc. In line 7, above, the value of N is a zero noun with left adjuncts in line 3 and right-adjunct in line 6.

Certain *wh*-strings (*who was there*, *which he saw*, etc.) give rise to another kind of null element. These strings consist of a *wh*-word followed by a Cl ASSERTION string (plus various adjuncts) from which one noun element has been "omitted," i.e. is said to be satisfied by a null element. In the output this is shown by writing ( ) under the omitted element. E.g. in sentence 7A (*The glucagon used throughout the structure work was a twice recrystallized lot (#208-158B-292A) prepared from hog pancreas that was homogeneous*), we have as right-adjunct of *pancreas* the string *that was homogeneous* which contains a Cl with omitted subject:

```
9.  C69 THAT Cl (Ø M)  =  XXX     Cl ASSERTION
                          THAT    8

8.  Cl ASSERTION       =  * SUBJECT  *  VERB  *  ØBJECT  R-V  *
                            ( )         WAS      7

7.  ADJECTIVE+         =  L-A   A                R-A
                                HOMOGENEOUS
```

Still another type of null element recognized by the present string grammar occurs in coordinate and comparative conjunctional strings (called Ql in the output). Here we identify the null element in the conjunctional string Ql with a correspondingly placed non-null element in the preceding (similarly structured) string to which Ql is conjoined; the interpretation is that the words of the preceding non-null element are repeated (in zero form, i.e. understood) in the conjunctional string. For example, in sentence 13 (*The hydrolyzates often contained traces of humin and had a faint blue-grey hue*), the subject of the conjoining assertion string in line 6 is considered to be a repetition (in zeroed form) of the subject of the string in line 4. This is indicated by writing in parentheses as the value of the subject in line 6 the words which satisfied the subject in line 4:

```
        SENTENCE     =  INTRODUCER    CENTER     M1        END-MARK
                                        4        AND 7        .

    4.  C1 ASSERTION =  * SUBJECT              * VERB         * OBJECT  R-V  *
                          1 HYDROLYZATES       2 CONTAINED      TRACES 3

    7.  Q1           =  CENTER
                        6

    6.  C1 ASSERTION =  * SUBJECT              * VERB   * OBJECT  R-V  *
                          (THE HYDROLYZATES)     HAD      5 HUE
```

Alternative analyses

For each sentence we show all the analyses of the sentence obtained by the machine. It seems surprising, then, that there are so few alternative analyses, most sentences having only one output parse. The explanation is simply that the program suppresses those analyses which are regularly predictable from a given analysis if one knows the main properties of the string grammar. 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*). If the first adjunct itself ends in N, then, unless there are further constraints to apply (e.g. subclass affinities), an ambiguity results, since the second adjunct can adjoin either the original N (repeatability) or the final N of the first adjunct (nesting). The most common situation is a sequence of P N's in r_N: N P N P N ... (*The house of my friend in Philadelphia* ...). However, the same type of ambiguity arises with subordinate conjunction strings and other sentence-adjuncts which can themselves have final sentence adjuncts: *I went there even though I was delayed because I had many things to do*.

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*.

r_V: *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. We should also be able to recognize certain P N's as sentence-adjuncts, such as those stating experimental conditions: *at 100°*, *for two hours*, etc. But additional work on subclassifications is necessary to establish these restrictions.

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 and nesting, the output prints the analysis due to nesting. Where strings in r_N, r_V and s.a. overlap, the output assigns the common string to the first adjunct set having permission to enter in the given situation (e.g. in line 10 of sentence 9, a P N after V is assigned to r_V, not to s.a.). Some of the outputs are particularly unhappy as a result of these conventions (e.g. in sentence 9, *hours* are *at 100 degrees*, *degrees* are *in vacuo*, etc.). The output conventions which suppress analyses can easily be dropped, or dropped selectively, if one prefers to display alternative analyses of this type, but the problem of correctly associating the adjuncts remains.

It should be mentioned that in all the sentences of the attached texts a third type of output suppression is in force. If a string is in an adjunct set and is also an object string, a second analysis due to a shift from adjunct to object may be possible. At present these alternative analyses are also not printed. Thus in sentence 3B (*Among them, methionine, tryptophan, valine and alanine are liberated from the C-terminus of the molecule by carboxypeptidase*), the verb *liberated* is taken as occurring with a zero passive object, and the sequence *from the C-terminus...* is taken as a right adjunct of the verb. Since we consider *from N* a possible passive object of *liberated*, another analysis - not printed - would show from the C-terminus... in the object position. (A genuine ambiguity is seen in They liberated the soldiers from considerations of policy.) Again, further restrictions involving subclasses might enable us to decide some cases. For example, it may be true that a P N in r_V (for certain prepositions) must contain a noun which is related to an adjective or a verb (from fear, from a feeling of responsibility, but not as r_V from his brother).

Alterations in the input sentences, due to lack of available space in the computer.

Glucagon Text

(1) Sentences 7 and 8 were each broken into two parts.

(2) In sentence 11, written below, we removed the eight words which are underlined (this does not affect the sentence-structure, though it may affect the meaning):

> For the study of amino acid composition, accurately weighed samples of crystalline glucagon were hydrolyzed for 20 or 72 hours in the absence of air at 107 to 109 degrees in about 200 volumes of 5.7 N HCl redistilled three times in glass.

(3) Sentence 8B was run in both full and shortened versions. Because of lack of computer space the full version succeeded only in finding two parses, after which the program halted for lack of space. However the shortened version obtained five parses (one redundant) which indicate the kind of parses the full version of the sentence would obtain if it had enough space in the computer to run to the end.

> Sentence 8B. The analyses show good over-all recovery, a complete absence of isoleucine, proline and cystine, and, in general, good stoichiometric relationships of the amino acid residues.

> Test Sentence 8B. The analyses show complete recovery, an absence of isoleucine, and good relationships.

Scientific American Text

(1) Sentence 14. Parenthetical expression eliminated.

(2) Sentence 16. The last four words on a dark ground were eliminated. Exactly the same structure had already been analyzed in the first half of the sentence:

> If it appears at first to show a dark cross on a light ground, it will in time shift and appear to show a light cross on a dark ground.

Additional Analyses

In the early version of the IPL program, the "freeze test," which is in effect only a device for saving computing time, was incorrectly applied to some of the sentences in the Glucagon text, and many parses were inadvertently suppressed in this fashion. The following is a list of those sentences from the Glucagon text to which this remark applies, together with an indication of the missing reading:

(2b) The (correct) analysis of *hydrazinolysis and carboxypeptidase treatment* in which *hydrazinolysis* is conjoined by the compound noun *carboxypeptidase treatment*;

(7b) The analysis in which *end group analysis and zone* is taken as an ℓ_N on *electrophoresis* (like *stone age copper and flint tools*).

(8a) The analysis of *below* as the object of *reported* (as in *The sailor was reported below*).

(14) The añalysis of *produced light straw-colored* as a repeated adjective on *hydrolyzates*.

Also a number of redundant readings due to the multibranched tree structure created by the parsing program have been ruled out:

14

1. Glucagon text, test sentence 8b, parse 5 (identical to parse 4);
2. *Scientific American* text, sentence 16, parse 1 (identical to parse 2).

A number of strings in the grammar are marked "rare," which means that they are not used unless no analysis of the sentence is possible when using a grammar which excludes them. The rare marker in the grammar ruled out the following additional readings (all from sentences of the Glucagon text):

(2b) An analysis in which the sentence is taken as two conjoined center strings: *The residue is threonine and carboxypeptidase (is) treatment*.

(7a) The analysis in which the center string is *Glucagon used a lot*, where the right adjunct on *used* is *throughout the structure*, and *structure* has the right adjunct *work was* (like *the fool I was*).

(14) The analysis in which the sequence *later crystalline preparations isolated in the presence of Versene produced light straw-colored hydrolyzates* is taken as the center string *preparations isolated hydrolyzates*, where *isolated* has the right adjunct *in the presence* and *presence* has the right adjunct *of Versene produced light* (like *of protein starved subjects*).

Other parses of sentence 16 of the *Scientific American* text were
15 not obtained because the program ran out of available space. In these
analyses *appears* has a zero object and *to show a dark cross* is a sentence
adjunct, i.e., *it appears (in order) to show*....

Short Form of the Output

From February 1966 on, two forms of output are produced by the FAP program: the long form described previously, in which the grammatical name of each successive string-element is printed along with its value in the sentence (as in the Glucagon and *Scientific American* texts); and a short form in which only the values of the elements are printed (as in the Metallurgy abstract). E.g. from the output of Sentence 2 of the Metallurgy abstract:*

Short form

```
2.  Cl ASSERTION  =  THEY   WERE   *3.   REDUCED 4.
```

Note that in the FAP program, sentence adjuncts are preceded by an asterisk, "*", to distinguish them from adjuncts of particular string elements, which are positionally indicated to the left or right of the element adjoined. Also, output-line numbers are distinguished from numbers which appear in the text by the appearance of a point "." to the right of an output-line number.

**Review of Metals Literature*, vol. 22, no. 8, August 1965.

Journal of the American Chemical Society, June 5, 1957

[CONTRIBUTION FROM THE LILLY RESEARCH LABORATORIES]

The Amino Acid Sequence of Glucagon. I. Amino Acid Composition and Terminal Amino Acid Analyses

BY W. W. BROMER, A. STAUB, E. R. DILLER, H. L. BIRD, L. G. SINN AND OTTO K. BEHRENS

RECEIVED NOVEMBER 19, 1956

Evidence is presented that glucagon is a small protein, consisting of a single chain of 29 amino acid residues. The N-terminal amino acid is histidine as determined by the dinitrophenylation method; the C-terminal residue is threonine on the basis of evidence obtained from hydrazinolysis and carboxypeptidase treatment. Glucagon contains single residues of 7 amino acids; among them, methionine, tryptophan, valine and alanine are liberated from the C-terminus of the molecule by carboxypeptidase.

Introduction

The preparation[1] in this Laboratory of a crystalline material of pancreatic origin, glucagon, has made available a model protein for fundamental biological and structural studies. This series of papers will present evidence leading to the elucidation of the complete amino acid sequence of glucagon. The present paper describes the quantitative amino acid analysis and other fundamental studies of primary importance to the sequence determination.

Experimental

Materials and Procedures.—The glucagon used throughout the structure work was a twice-recrystallized lot (#208-158B-292A) prepared from hog pancreas that was homogeneous according to end group analysis and zone electrophoresis on starch.[1] The high purity of the sample is also demonstrated by the analyses reported below which show good over-all recovery, a complete absence of isoleucine, proline and cystine, and, in general, good stoichiometric relationships of the amino acid residues.

Moisture content of the preparation was 10.9% as determined by drying for 24 hr. at 100° *in vacuo* over P_2O_5. The nitrogen content was 17.45% (Kjeldahl).

For the study of amino acid composition, accurately weighed samples of crystalline glucagon were hydrolyzed for 20 or 72 hr. in the absence of air at 107 to 109° in about 200 volumes of 5.7 *N* HCl redistilled three times in glass. The hydrolyzates were processed in a manner similar to that described by Smith and Stockell.[2] The hydrolyzates often contained traces of humin and had a faint blue-gray hue. The presence of trace metals in the crystalline preparation may have catalyzed the destruction of tryptophan since later crystalline preparations isolated in the presence of Versene produced light straw-colored hydrolyzates. Humin formation did not hamper the handling of the sample and probably did not affect the results.

Amino Acid Determination.—The comparatively rapid Levy[3] method, based on the paper chromatographic separation of the dinitrophenyl- (DNP-) amino acids, was employed initially in the work and was followed by the more rigorous Dowex 50 column chromatographic method of Moore and Stein.[4] The dinitrophenylation method furnished adequate preliminary data to support the structure work and, as will be reported, the two methods gave similar results.

The Dinitrophenylation Method.—Throughout this work dinitrophenylations were performed for 90 min. at 40°, *p*H 9.0, essentially following the Levy modification[3] of the method of Sanger.[5] Prior to the paper chromatographic separation of the DNP-amino acids, dinitrophenol was removed by sublimation according to the method described by Mills.[6] DNP-derivatives were identified by comparison with known DNP-amino acids.[7]

Only 20 hr. hydrolyzates were used with the Levy method since no DNP-peptides were observed. Recognizing that the correction factors published by Levy for the destruction of the amino acids in insulin may not be completely applicable to another protein handled under different conditions, new factors were determined for glucagon. This was accomplished by utilizing the insulin factors as a first approximation. From these data a synthetic amino acid mixture resembling glucagon was prepared, hydrolyzed and treated identically to a protein hydrolyzate. In this manner new destruction factors were calculated. This process was repeated, giving a more precise set of correction factors which were used in the calculation of the number of amino acid residues in glucagon. The correction factors are as follows: aspartic acid, 1.10; threonine, 1.22; serine, 1.32; glutamic acid, 1.20; glycine, 1.33; alanine, 1.28; valine, 1.20; methionine, 2.12; tyrosine, 1.60; phenylalanine, 1.20; histidine, 2.19; lysine, 0.70; and arginine, 1.38. Of interest is the finding that, in every instance, the factors are higher than those reported by Levy, averaging about 17% greater. Histidine and methionine particularly appear to be subject to greater losses than previously reported.[3]

Table I indicates the results of the preliminary amino acid analysis by the Levy method. With the relatively few analyses it is not surprising that the tyrosine and arginine values do not fall near whole-residue figures. Di-DNP-tyrosine analyses at best are quite variable,[3] and DNP-arginine, being water-soluble, is rather difficult to desalt completely prior to paper chromatography. Indeed, the water-soluble DNP-derivatives of histidine, lysine and arginine have generally proved to be difficult to handle. Ex-

TABLE I

AMINO ACID ANALYSIS OF GLUCAGON

Amino acid	Av. quantity found,[a] μM	Calcd. no. of amino acid residues[b]	No. of amino acid residues to nearest integer
Aspartic acid	0.34	4.0	4
Threonine	.24	2.8	3
Serine	.30	3.5	4[c]
Glutamic acid	.24	2.8	3
Glycine	.09	1.1	1
Alanine	.09	1.1	1
Valine	.08	0.9	1
Methionine	.06	.7	1
Leucine	.18	2.1	2
Tyrosine	.13	1.5	1 or 2
Phenylalanine	.17	2.0	2
Histidine	.11	1.3	1
Lysine	.08	0.9	1
Arginine	.22	2.6	2 or 3
Tryptophan[d]	.085	1.0	1
			28–30

[a] Average of 2 or 3 analyses with exception of single glycine determination. [b] On basis of 7 amino acids being present as single residues. (0.085 μM is average of 7.) [c] Since serine is subjected to approximately 10% destruction during acid hydrolysis (*cf.* Table II), 4 is a more reasonable figure than 3. [d] Chemical analysis.[8]

[1] A. Staub, L. Sinn and Otto K. Behrens, *J. Biol. Chem.*, **214**, 619 (1955).

[2] E. L. Smith and A. Stockell, *ibid.*, **207**, 501 (1954).

[3] A. L. Levy, *Nature*, **174**, 126 (1954).

[4] S. Moore and W. H. Stein, *J. Biol. Chem.*, **192**, 663 (1951).

[5] F. Sanger, *Biochem. J.*, **39**, 507 (1945).

[6] G. L. Mills, *ibid.*, **50**, 707 (1952).

[7] Authentic DNP-amino acids were kindly supplied by Dr. Hans Neurath, Univ. of Washington, School of Medicine.

[8] J. R. Spies and D. C. Chambers, *Anal. Chem.*, **20**, 30 (1948).

JAN 29 1965

```
PARSE NØ. 1

SENTENCE 1. EVIDENCE IS PRESENTED THAT GLUCAGON IS A SMALL PROTEIN,
CONSISTING OF A SINGLE CHAIN OF 29 AMINO ACID RESIDUES.


 SENTENCE           = INTRØDUCER  CENTER  END-MARK
                                  10      .

10. C1 ASSERTIØN    = *  SUBJECT     *  VERB  *  ØBJECT  R-V  *
                         EVIDENCE       IS       1            9

 1. C132 PASSIVE    = VEN         *  PASSIVE-Ø  R-V  *
                      PRESENTED

 9. C108            = THAT  C1 ASSERTIØN
                      THAT  8

 8. C1 ASSERTIØN    = *  SUBJECT     *  VERB  *  ØBJECT              R-V  *
                         GLUCAGØN       IS       2 PRØTEIN , 7

 2. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                      A                    SMALL

 7. 0 VING+ØBJ      = VING        *  ØBJECT  R-V  *
                      CØNSISTING     6

 6. C20 P N         = L-P  P   N
                           ØF  3 CHAIN 5

 3. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                      A                    SINGLE

 5. C20 P N         = L-P  P   N
                           ØF  4 RESIDUES

 4. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                               29          AMINØ               ACID
```

FEB. 5 1965

```
   SENTENCE 2A. THE N-TERMINAL AMINO ACID IS HISTIDINE
   AS DETERMINED BY THE DINITROPHENYLATION METHOD.

   SENTENCE        = INTRØDUCER  CENTER   END-MARK
                                  6       SEMICØLØN

6. CI ASSERTIØN    =  *   SUBJECT    *  VERB     *   ØBJECT      R-V    *
                          1 ACID        IS           HISTIDINE        5

1. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE          TYPE NS  NØUN
                     THE                  N-TERMINAL AMINØ

5. C162            = L-CS  CS2  C132 PASSIVE
                           AS   4

4. C132 PASSIVE    = VEN               *  PASSIVE-Ø  R-V    *
                     DETERMINED 3

3. C20 P N         = L-P  P    N
                          BY   2 METHØD

2. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE                                      DINITRØPHENYLATIØN
```

FEB. 5 1965

```
SENTENCE 2 B. THE C-TERMINAL RESIDUE IS THREONINE ON THE
BASIS OF EVIDENCE OBTAINED FROM HYDRAZINOLYSIS AND
CARBOXYPEPTIDASE TREATMENT .

SENTENCE         = INTRØDUCER  CENTER  END-MARK
                               9       .

9. CI ASSERTIØN  = *  SUBJECT      *  VERB  *  ØBJECT     R-V  *
                      1 RESIDUE       IS       THREØNINE    8

1. L-N           = ARTICLE  QUANTIFIER  ADJECTIVE   TYPE NS  NØUN
                   THE                  C-TERMINAL

8. C20 P N       = L-P  P    N
                        ØN   2 BASIS 7

2. L-N           = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                   THE

7. C20 P N       = L-P  P    N
                        ØF   EVIDENCE 6

6. C132 PASSIVE  = VEN            *  PASSIVE-Ø  R-V  *
                   ØBTAINED 5

5. C20 P N       = L-P  P     N
                        FRØM  4 TREATMENT

4. L-N           = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN             M1
                                                            HYDRAZINØLYSIS   AND 3

3. Q1            = NØUN
                   CARBØXYPEPTIDASE
```

```
PARSE 1

SENTENCE 3 A. GLUCAGON CONTAINS SINGLE RESIDUES OF 7 AMINO ACIDS .


   C                    = INTRØDUCER  CENTER  END-MARK
                                      4       SEMICØLØN

4. C1 ASSERTIØN         =  *  SUBJECT      *  VERB        *  ØBJECT          R-V   *
                              GLUCAGØN        CØNTAINS       1 RESIDUES 3

1. L-N                  = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                               SINGLE

3. C20 P N              = L-P  P    N
                               ØF   2 ACIDS

2. L-N                  = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                   7           AMINØ
```

FEB. 4 1965

```
SENTENCE 3B. AMONG THEM,METHIONINE,TRYPTOPHAN ,VALINE AND ALANINE ARE
LIBERATED FROM THE C-TERMINUS OF THE MOLECULE BY CARBOXYPEPTIDASE.


   SENTENCE           = INTRØDUCER  CENTER  END-MARK
                                    11       .


11. C1 ASSERTIØN      =  *   SUBJECT               *  VERB    *  ØBJECT   R-V    *
                         1   METHIØNINE . 4           ARE        10


 1. C20 P N           = L-P  P         N
                             AMØNG     THEM .


 4. Q1                = A20            M10
                        TRYPTØPHAN      . 3


10. C132 PASSIVE      = VEN                 *  PASSIVE-Ø   R-V    *
                        LIBERATED 9


 3. Q1                = A20        M1
                        VALINE     AND 2


 9. C20 P N           = L-P  P       N
                             FRØM    5 C-TERMINUS 8


 2. Q1                = A20
                        ALANINE


 5. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THE


 8. C20 P N           = L-P  P     N
                             ØF    6 MØLECULE 7


 6. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THE


 7. C20 P N           = L-P  P     N
                             BY    CARBØXYPEPTIDASE
```

FEB 11 1965

```
SENTENCE 4.  THE PREPARATION IN THIS LABORATORY OF A CRYSTALLINE MATERIAL
OF PANCREATIC ORIGIN, GLUCAGON, HAS MADE AVAILABLE A MODEL PROTEIN FOR
FUNDAMENTAL BIOLOGICAL AND STRUCTURAL STUDIES.


    SENTENCE          = INTRØDUCER  CENTER  END-MARK
                                    15        .


15. C1 ASSERTIØN      =  *  SUBJECT                *  VERB    *  ØBJECT  R-V   *
                            1 PREPARATIØN 7           HAS        14


 1. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THE


 7. C20 P N           = L-P  P    N
                             IN   2 LABØRATØRY 6


14. C131              = VEN      *  ØBJECT  R-V   *
                        MADE        13


 2. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THIS


 6. C20 P N           = L-P  P    N
                             ØF   3 MATERIAL 5


13. C188              = ADJECTIVE+    *  N
                        8                9 PRØTEIN 12


 3. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE    TYPE NS  NØUN
                        A                    CRYSTALLINE


 5. C20 P N           = L-P  P    N
                             ØF   4 ØRIGIN , GLUCAGØN ,


 8. ADJECTIVE+        = L-A  A           R-A
                             AVAILABLE


 9. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        A                                        MØDEL


12. C20 P N           = L-P  P     N
                             FØR   11 STUDIES


 4. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE   TYPE NS  NØUN
                                             PANCREATIC


11. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE                M1        TYPE NS  NØUN
                                             FUNDAMENTAL BIØLØGICAL   AND 10


10. Q1                = ADJECTIVE
                        STRUCTURAL
```

JAN 27 1965

```
PARSE 1
SENTENCE 5. THIS SERIES OF PAPERS WILL PRESENT EVIDENCE LEADING TO THE
ELUCIDATION OF THE COMPLETE AMINO ACID SEQUENCE OF GLUCAGON.


  SENTENCE          = INTRØDUCER  CENTER   END-MARK
                                   9        .

9. C1 ASSERTIØN     =  *  SUBJECT         *  VERB               *  ØBJECT         R-V     *
                          1 SERIES 2         WILL PRESENT          EVIDENCE 8

1. L-N              = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                      THIS

2. C20 P N          = L-P   P     N
                            ØF    PAPERS

8.   VING+ØBJ       = VING           *  ØBJECT  R-V     *
                      LEADING 7

7. C20 P N          = L-P   P     N
                            TØ    3 ELUCIDATIØN 6

3. L-N              = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                      THE

6. C20 P N          = L-P   P     N
                            ØF    4 SEQUENCE 5

4. L-N              = ARTICLE   QUANTIFIER   ADJECTIVE          TYPE NS   NØUN
                      THE                    CØMPLETE AMINØ               ACID

5. C20 P N          = L-P   P     N
                            ØF    GLUCAGØN
```

```
    SENTENCE 6. THE PRESENT PAPER DESCRIBES THE QUANTITATIVE AMINO ACID
    ANALYSIS AND OTHER FUNDAMENTAL STUDIES OF PRIMARY IMPORTANCE TO THE
    SEQUENCE DETERMINATION.


    SENTENCE       = INTRØDUCER  CENTER  M1        END-MARK
                                  3      AND 10    .


 3. C1 ASSERTIØN   =  *  SUBJECT     *  VERB          *  ØBJECT       R-V    *
                         1 PAPER        DESCRIBES        2 ANALYSIS


10. Q1 CØNJUNCT    = CENTER
                     9

 1. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE   TYPE NS  NØUN
                     THE                  PRESENT

 2. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE             TYPE NS  NØUN
                     THE                  QUANTITATIVE AMINØ             ACID

 9. C1 ASSERTIØN   =  *  SUBJECT                    *  VERB              .*  ØBJECT         R
                         ( THE PRESENT PAPER )         ( DESCRIBES )         4 STUDIES 8

 4. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE            TYPE NS  NØUN
                                          ØTHER FUNDAMENTAL

 8. C20 P N        = L-P  P    N
                          ØF   5 IMPØRTANCE 7

 5. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                          PRIMARY

 7. C20 P N        = L-P  P    N
                          TØ   6 DETERMINATIØN

 6. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE                                      SEQUENCE
```

JAN. 26 1965

```
PARSE 1

SENTENCE 7 A. THE GLUCAGON USED THROUGHOUT THE STRUCTURE
WORK WAS A TWICE-RECRYSTALLIZED LOT( 208-158B-292A)
PREPARED FROM HOG PANCREAS THAT WAS HOMOGENOUS.


 SENTENCE         = INTRØDUCER  CENTER  END-MARK
                                12         .

                                                                                     R-V
12. C1 ASSERTIØN    =  *   SUBJECT          *  VERB    *   ØBJECT
                           1 GLUCAGØN 4        WAS         5 LØT ( 208-158B-292A ) 11

 1. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                      THE

 4. C132 PASSIVE    = VEN        *  PASSIVE-Ø  R-V    *
                      USED 3

 5. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE             TYPE NS  NØUN
                      A                    TWICE-RECRYSTALLIZED

11. C132 PASSIVE    = VEN              *  PASSIVE-Ø  R-V    *
                      PREPARED 10

 3. C20 P N         = L-P  P           N
                           THRØUGHØUT  2 WØRK

10. C20 P N         = L-P  P       N
                           FRØM    6 PANCREAS 9

 2. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                      THE                                      STRUCTURE

 6. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                               HØG

 9. C69 THAT C1(ØM)= XXX    C1 ASSERTIØN
                     THAT   8

 8. C1 ASSERTIØN    =  *  SUBJECT   *  VERB   *  ØBJECT  R-V   *
                          ( )          WAS       7

 7. ADJECTIVE+      = L-A  A          R-A
                           HØMØGENØUS
```

```
PARSE NØ. 1                                                   FEB 4  1965

SENTENCE 7 B. IT WAS HOMOGENOUS ACCORDING-TO END
GROUP ANALYSIS AND ZONE ELECTROPHORESIS ON STARCH.

SENTENCE             = INTRØDUCER  CENTER  END-MARK
                                   7       .

7. C1 ASSERTIØN      =  *  SUBJECT   *  VERB   *  ØBJECT  R-V   *
                           IT           WAS       1             6

1. ADJECTIVE+        = L-A  A                 R-A
                            HØMØGENØUS

6. C20 P N           = L-P  P               N             M1
                            ACCØRDING-TØ    2 ANALYSIS    AND 5

2. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                                END GRØUP

5. Q1                = N
                       3 ELECTRØPHØRESIS 4

3. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                                ZØNE

4. C20 P N           = L-P  P    N
                            ØN   STARCH
```

JAN. 29 1965

```
PARSE 1

SENTENCE 8 A. THE HIGH PURITY OF THE SAMPLE IS ALSO
DEMONSTRATED BY THE ANALYSES REPORTED BELOW.


SENTENCE          = INTRØDUCER  CENTER  END-MARK
                                10      .


10. C1 ASSERTIØN  = *  SUBJECT        *  VERB   *  ØBJECT  R-V   *
                       1 PURITY 3        IS     4  9


 1. L-N           = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                    THE                  HIGH


 3. C20 P N       = L-P  P    N
                         ØF   2 SAMPLE


 4. B16           = ADVERB
                    ALSØ


 9. C132 PASSIVE  = VEN                   *  PASSIVE-Ø  R-V   *
                    DEMØNSTRATED 8


 2. L-N           = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                    THE


 8. C20 P N       = L-P  P    N
                         BY   5 ANALYSES 7


 5. L-N           = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                    THE


 7. C132 PASSIVE  = VEN           *  PASSIVE-Ø  R-V   *
                    REPØRTED   6


 6. B16           = ADVERB
                    BELØW
```

MAR. 5 1965

```
TEST SENTENCE 8B.  THE ANALYSES SHOW COMPLETE RECOVERY,
AN ABSENCE OF ISOLEUCINE, AND GOOD RELATIONSHIPS.
PARSE 1
SENTENCE           = INTRØDUCER  CENTER   END-MARK
                                 8        .

8. C1 ASSERTIØN    =  *  SUBJECT       *  VERB     *  ØBJECT              R-V
                         1 ANALYSES       SHØW        2 RECØVERY , 7

1. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE

2. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                          CØMPLETE

7. APPØSITIØN      = C91
                     3 ABSENCE 6

3. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     AN

6. C20 P N         = L-P  P    N              M1
                          ØF   ISØLEUCINE ,   AND 5

5. Q1 CØNJUNCT     = N
                     4 RELATIØNSHIPS

4. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                          GØØD
```

MAR 5 1965

```
TEST SENTENCE 8B.  THE ANALYSES SHOW COMPLETE RECOVERY,
AN ABSENCE OF ISOLEUCINE, AND GOOD RELATIONSHIPS.

PARSE 2
SENTENCE          = INTRØDUCER   CENTER   M1       END-MARK
                                 6        AND 9    .

6. C1 ASSERTIØN   = *  SUBJECT        *  VERB     *  ØBJECT              R-V    *
                       1 ANALYSES        SHØW        2 RECØVERY , 5

9. Q1 CØNJUNCT    = CENTER
                    8

1. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                    THE

2. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                           CØMPLETE

5. APPØSITIØN     = C91
                    3 ABSENCE 4

8. C1 ASSERTIØN   = *  SUBJECT              *  VERB        *  ØBJECT              R-V    *
                       ( THE ANALYSES )        ( SHØW )       7 RELATIØNSHIPS

3. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                    AN

4. C20 P N        = L-P   P    N
                          ØF   ISØLEUCINE ,

7. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                           GØØD
```

MAR 5 1965

```
TEST SENTENCE 8B.  THE ANALYSES SHOW COMPLETE RECOVERY,
AN ABSENCE OF ISOLEUCINE, AND GOOD RELATIONSHIPS.

PARSE 3

SENTENCE             = INTRØDUCER   CENTER   M10    END-MARK
                                    3        , 9    .

3. C1 ASSERTIØN      =  *   SUBJECT          *  VERB     *  ØBJECT        R-V    *
                            1 ANALYSES          SHØW        2 RECØVERY

9. Q1 CØNJUNCT       = CENTER
                       8

1. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                       THE

2. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                              CØMPLETE

8. C1 ASSERTIØN      =  *   SUBJECT              *  VERB         *  ØBJECT          R-V    *
                            ( THE ANALYSES )        ( SHØW )        4 ABSENCE 7

4. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                       AN

7. C20 P N           = L-P   P     N                M1
                             ØF    ISØLEUCINE ,     AND 6

6. Q1 CØNJUNCT       = N
                       5 RELATIØNSHIPS

5. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                              GØØD
```

MAR. 5 1965

```
    TEST SENTENCE 8B.  THE ANALYSES SHOW COMPLETE RECOVERY,
    AN ABSENCE OF ISOLEUCINE, AND GOOD RELATIONSHIPS.
    PARSE 4 (PREFERRED)
    SENTENCE             = INTRØDUCER  CENTER  M10    END-MARK
                                       3       . 10   .

 3. C1 ASSERTIØN         = *  SUBJECT        *  VERB    *  ØBJECT        R-V   *
                              1 ANALYSES        SHØW       2 RECØVERY

10. Q1 CØNJUNCT          = CENTER  M1
                           6       AND 9

 1. L-N                  = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                           THE

 2. L-N                  = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                CØMPLETE

 6. C1 ASSERTIØN         = *  SUBJECT             *  VERB        *  ØBJECT           R-V   *
                              ( THE ANALYSES )       ( SHØW )       4 ABSENCE 5

 9. Q1 CØNJUNCT          = CENTER
                           8

 4. L-N                  = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                           AN

 5. C20 P N              = L-P  P     N
                                ØF    ISØLEUCINE ,

 8. C1 ASSERTIØN         = *  SUBJECT             *  VERB        *  ØBJECT              R-V
                              ( THE ANALYSES )       ( SHØW )       7 RELATIØNSHIPS

 7. L-N                  = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                GØØD
```

MAR. 5 1965

```
TEST SENTENCE 8B.  THE ANALYSES SHOW COMPLETE RECOVERY,
AN ABSENCE OF ISOLEUCINE, AND GOOD RELATIONSHIPS.

PARSE 5 (REDUNDANT)

SENTENCE          = INTRØDUCER   CENTER   M10    M1       END-MARK
                                 3        , 7    AND 10   .


3. C1 ASSERTIØN   =  *  SUBJECT        *  VERB    *  ØBJECT        R-V    *
                        1 ANALYSES        SHØW       2 RECØVERY


7. Q1 CØNJUNCT    = CENTER
                    6


10. Q1 CØNJUNCT   = CENTER
                    9


1. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                    THE


2. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                           CØMPLETE


6. C1 ASSERTIØN   =  *  SUBJECT            *  VERB       *  ØBJECT           R-V    *
                        ( THE ANALYSES )      ( SHØW )      4 ABSENCE 5


9. C1 ASSERTIØN   =  *  SUBJECT            *  VERB       *  ØBJECT                  R-V    *
                        ( THE ANALYSES )      ( SHØW )      8 RELATIØNSHIPS


4. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                    AN


5. C20 P N        = L-P   P    N
                          ØF   ISØLEUCINE ,


8. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                           GØØD
```

SENTENCE 8 B.THE ANALYSES SHOW GOOD OVER-ALL RECOVERY, A COMPLETE ABSENCE OF ISOLEUCINE,PROLINE AND CYSTINE,AND,IN GENERAL,GOOD STOICHIOMETRIC RELATIONSHIPS OF THE AMINO ACID RESIDUES.

```
   PARSE 1

   SENTENCE          = INTRØDUCER  CENTER  END-MARK
                                   13        .


13. C1 ASSERTIØN     =  *  SUBJECT        *  VERB     *  ØBJECT                R-V   *
                           1 ANALYSES        SHØW        2 RECØVERY . 12

 1. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                       THE

 2. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE        TYPE NS  NØUN
                                            GØØD ØVER-ALL

12. APPØSITIØN       = C91
                       3 ABSENCE 11

 3. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                       A                    CØMPLETE

11. C20 P N          = L-P  P    N                  M1
                            ØF   ISØLEUCINE . 5     AND . 6 10

 5. Q1 CØNJUNCT      = NØUN      M1
                       PROLINE   AND  4


 6. C19              = P    ADJECTIVE  M10
                       IN   GENERAL     .

10. Q1 CØNJUNCT      = N
                       7 RELATIØNSHIPS 9

 4. Q1 CØNJUNCT      = NØUN      M10
                       CYSTINE    .

 7. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE                  TYPE NS  NØUN
                                            GØØD STØICHIØMETRIC

 9. C20 P N          = L-P  P    N
                            ØF   8 RESIDUES

 8. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                       THE                  AMINØ               ACID
```

MAR 5 1965

```
SENTENCE 8 B.THE ANALYSES SHOW GOOD OVER-ALL RECOVERY, A COMPLETE ABSENCE
OF ISOLEUCINE,PROLINE AND CYSTINE,AND,IN GENERAL,GOOD STOICHIOMETRIC
RELATIONSHIPS OF THE AMINO ACID RESIDUES.
PARSE 2
SENTENCE          = INTRØDUCER   CENTER    M1             END-MARK
                                 8         AND , 9 14     .


 8. C1 ASSERTIØN     =  *   SUBJECT        *   VERB     *   ØBJECT               R-V    *
                            1 ANALYSES         SHØW         2 RECØVERY , 7


 9. C19               = P     ADJECTIVE   M10
                        IN    GENERAL      .


14. Q1 CØNJUNCT       = CENTER
                        13


 1. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                        THE


 2. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE        TYPE NS   NØUN
                                               GØØD ØVER-ALL


 7. APPØSITIØN        = C91
                        3 ABSENCE 6


13. C1 ASSERTIØN    =  *   SUBJECT                *   VERB        *   ØBJECT                     R-V    *
                           ( THE ANALYSES )           ( SHØW )        10 RELATIØNSHIPS 12


 3. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                        A                      CØMPLETE


 6. C20 P N           = L-P   P     N
                              ØF    ISØLEUCINE , 5


10. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE               TYPE NS   NØUN
                                               GØØD STØICHIØMETRIC


12. C20 P N           = L-P   P     N
                              ØF    11 RESIDUES


 5. Q1 CØNJUNCT       = NØUN       M1
                        PRØLINE    AND 4


11. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                        THE                    AMINØ                 ACID



 4. Q1 CØNJUNCT       = NØUN       M10
                        CYSTINE    .
```

```
    SENTENCE 9. MOISTURE CONTENT OF THE PREPARATION WAS 10.9 PERCENT AS
    DETERMINED BY DRYING FOR 24 HOURS AT 100 DEGREES IN VACUO OVER P2O5.


    SENTENCE          = INTRØDUCER  CENTER   END-MARK
                                    15          .

15. C1 ASSERTIØN      =  *   SUBJECT          *   VERB    *   ØBJECT          R-V    *
                             1 CØNTENT 3          WAS         10.9 PERCENT            14

 1. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                                 MØISTURE

 3. C20 P N           = L-P   P     N
                              ØF    2 PREPARATIØN

14. C162              = L-CS  CS2  C132 PASSIVE
                              AS    13

 2. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THE

13. C132 PASSIVE      = VEN              *   PASSIVE-Ø  R-V    *
                        DETERMINED   12

12. C164              = L-CS  CS4   0
                              BY    11

11. C81 VING STG      = ARTICLE  C133 VING STG
                                 10

10. C133 VING STG     = VING          *   ØBJECT  R-V    *
                        DRYING 9

 9. C20 P N           = L-P   P     N
                              FØR   4 HØURS 8

 4. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                 24

 8. C20 P N           = L-P   P     N
                              AT    5 DEGREES 7

 5. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                 100

 7. C20 P N           = L-P   P     N
                              IN    VACUØ 6

 6. C20 P N           = L-P   P      N
                              ØVER   P2Ø5
```

```
                                                  FEB 8   1965
    PARSE NØ. 1

   SENTENCE 10. THE NITROGEN CONTENT WAS 17.45 PERCENT (KJELDAHL).


    SENTENCE            = INTRØDUCER   CENTER   END-MARK
                                         3         .


3. C1 ASSERTIØN         =  *   SUBJECT        *   VERB     *   ØBJECT   R-V    *
                               1 CØNTENT          WAS          2


1. L-N                  = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                          THE                                          NITRØGEN


2. ADJECTIVE+           = L-Q   B14                              R-Q
                                17.45 PERCENT ( KJELDAHL )
```

MAR. 3 1966

```
SENTENCE 11 - REDUCED.  FOR THE STUDY OF AMIÑO ACID COMPOSITION,
ACCURATELY WEIGHED SAMPLES WERE HYDROLYZED FOR 20 OR 72 HOURS IN
1              AIR AT 107 TO 109 DEGREES IN ABOUT 200 VOLUMES OF
5.7 N HCL REDISTILLED THREE TIMES.


    SENTENCE           = INTRØDUCER  CENTER  END-MARK
                                     21      .

21. C1 ASSERTIØN   =  *  SUBJECT      *  VERB    *  ØBJECT  R-V   *
                     4   6 SAMPLES       WERE       20

 4. C20 P N        = L-P  P    N
                          FØR  1 STUDY 3

 6. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                          5 WEIGHED

20. C132 PASSIVE   = VEN              *  PASSIVE-Ø  R-V   *
                     HYDRØLYZED 19

 1. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE

 3. C20 P N        = L-P  P    N
                          ØF   2 CØMPØSITIØN ,

 5. B16            = ADVERB
                     ACCURATELY

19. C20 P N        = L-P  P    N
                          FØR  8 HØURS 18

 2. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                          AMINØ               ACID

 8. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                              20 ØR 7

18. C20 P N        = L-P  P    N
                          IN   AIR 17

 7. Q1 CØNJUNCT    = QUANTIFIER
                     72

17. C20 P N        = L-P  P    N
                          AT   10 DEGREES 16
```

SENTENCE 11, P. 2

```
10. L-N                = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                   107 TØ 9

16. C20 P N            = L-P   P      N
                               IN     12 VØLUMES 15

 9. Q1 CØNJUNCT        = QUANTIFIER
                         109

12. L-N                = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                   11 200

15. C20 P N            = L-P   P      N
                               ØF     13 HCL 14

11. B16                = ADVERB
                         ABØUT

13. L-N                = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                                5.7 N

14. C132 PASSIVE       = VEN                *   PASSIVE-Ø   R-V                *
                         REDISTILLED                        THREE TIMES
```

FEB. 5 1965

```
    SENTENCE 12.  THE HYDROLYZATES WERE PROCESSED IN A MANNER SIMILAR TO
    THAT DESCRIBED BY SMITH AND STOCKFLL.


    SENTENCE          = INTRØDUCER  CENTER  END-MARK
                                    11       .


11. CI ASSERTIØN      =  *  SUBJECT              *  VERB     *  ØBJECT  R-V    *
                            1 HYDRØLYZATES          WERE        10

 1. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THE

10. C132 PASSIVE      = VEN             *  PASSIVE-Ø  R-V    *
                        PRØCESSED 9

 9. C20 P N           = L-P  P    N
                             IN   2 MANNER 8

 2. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        A

 8. ADJECTIVE+        = L-A  A          R-A
                             SIMILAR    7

 7. C20 P N           = L-P  P    N
                             TØ   3 6

 3. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                        THAT

 6. C132 PASSIVE      = VEN             *  PASSIVE-Ø  R-V    *
                        DESCRIBED 5

 5. C20 P N           = L-P  P    N
                             BY   SMITH AND 4

 4. Q1                = A20
                        STØCKELL
```

FEB. 3 1965

```
NØ MØRE PARSES.
        SENTENCE 13. THE HYDROLYZATES OFTEN CONTAINED TRACES OF HUMIN AND HAD A
        FAINT BLUE-GRAY HUE.


      SENTENCE            = INTRØDUCER  CENTER  M1       END-MARK
                                        4       AND 7    .


    4. C1 ASSERTIØN       =  *  SUBJECT               *  VERB             *  ØBJECT     R-V   *
                                1 HYDRØLYZATES     2   CØNTAINED           TRACES 3

    7. Q1                 = CENTER
                            6

    1. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                            THE

    2. R16                = ADVERB
                            ØFTEN

    3. C20 P N            = L-P  P    N
                                 ØF   HUMIN

    6. C1 ASSERTIØN       =  *  SUBJECT                  *  VERB   *  ØBJECT  R-V   *
                                ( THE HYDRØLYZATES )        HAD       5 HUE

    5. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE          TYPE NS  NØUN
                            A                    FAINT BLUE-GRAY
```

MAR 5 1965

```
SENTENCE 14.  THE PRESENCE OF TRACE METALS IN THE CRYSTALLINE PREPARATION
MAY HAVE CATALYZED THE DESTRUCTION OF TRYPTOPHAN SINCE LATER CRYSTALLINE
PREPARATIONS ISOLATED IN THE PRESENCE OF VERSENE PRODUCED LIGHT STRAW-
COLORED HYDROLYZATES.

 PARSE 1

 SENTENCE          = INTRØDUCER  CENTER  END-MARK
                                 19      .

19. C1 ASSERTIØN   =  *  SUBJECT           *  VERB         *  ØBJECT  R-V   *
                         1 PRESENCE 5         MAY HAVE        8             18

 1. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE

 5. C20 P N        = L-P  P    N
                          ØF   2 METALS 4

 8. C131           = VEN          *  ØBJECT              R-V
                     CATALYZED       6 DESTRUCTIØN 7

18. C161           = L-CS  CS1     C1 ASSERTIØN
                           SINCE   17

 2. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                              TRACE

 4. C20 P N        = L-P  P    N
                          IN   3 PREPARATIØN

 6. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE

 7. C20 P N        = L-P  P    N
                          ØF   TRYPTØPHAN

17. C1 ASSERTIØN   =  *  SUBJECT                  *  VERB     *  ØBJECT             R-V    *
                     9   10 PREPARATIØNS 15          LIGHT       16 HYDRØLYZATES

 3. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE     TYPE NS  NØUN
                     THE                  CRYSTALLINE

 9. B16            = ADVERB
                     LATER

10. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE     TYPE NS  NØUN
                                          CRYSTALLINE

15. C132 PASSIVE   = VEN            *  PASSIVE-Ø  R-V   *
                     ISØLATED 14

16. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE       TYPE NS  NØUN
                                          STRAW-CØLØRED

14. C20 P N        = L-P  P    N
                          IN   11 PRESENCE 13

11. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                     THE

13. C20 P N        = L-P  P    N
                          ØF   VERSENE 12

12. C132 PASSIVE   = VEN         *  PASSIVE-Ø  R-V   *
                     PRØDUCED
```

```
SENTENCE 14.  THE PRESENCE OF TRACE METALS IN THE CRYSTALLINE PREPARATION
MAY HAVE CATALYZED THE DESTRUCTION OF TRYPTOPHAN SINCE LATER CRYSTALLINE
PREPARATIONS ISOLATED IN THE PRESENCE OF VERSENE PRODUCED LIGHT STRAW-
COLORED HYDROLYZATES.

     SENTENCE          = INTRØDUCER  CENTER  END-MARK
     PARSE 2                         18        .

18.  C1 ASSERTIØN      =  *  SUBJECT          *  VERB       *  ØBJECT  R-V   *
                             1 PRESENCE 5        MAY HAVE      8             17

 1.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                         THE

 5.  C20 P N           = L-P  P     N
                              ØF    2 METALS 4

 8.  C131              = VEN          *  ØBJECT              R-V
                         CATALYZED       6 DESTRUCTIØN 7

17.  C161              = L-CS  CS1     C1 ASSERTIØN
                               SINCE   16

 2.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                                  TRACE

 4.  C20 P N           = L-P  P     N
                              IN    3 PREPARATIØN

 6.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                         THE

 7.  C20 P N           = L-P  P     N
                              ØF    TRYPTØPHAN

16.  C1 ASSERTIØN      =  *  SUBJECT                *  VERB       *  ØBJECT             R-V   *
                         9   10 PREPARATIØNS 14        PRØDUCED      15 HYDRØLYZATES

 3.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE    TYPE NS  NØUN
                         THE                  CRYSTALLINE

 9.  D16               = ADVERB
                         LATER

10.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE    TYPE NS  NØUN
                                              CRYSTALLINE

14.  C132 PASSIVE      = VEN            *  PASSIVE-Ø  R-V   *
                         ISØLATED 13

15.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE              TYPE NS  NØUN
                                              LIGHT STRAW-CØLØRED

13.  C20 P N           = L-P  P     N
                              IN    11 PRESENCE 12

11.  L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                         THE

12.  C20 P N           = L-P  P     N
                              ØF    VERSENE
```

MAR. 5 1965

SENTENCE 14. THE PRESENCE OF TRACE METALS IN THE CRYSTALLINE PREPARATION MAY HAVE CATALYZED THE DESTRUCTION OF TRYPTOPHAN SINCE LATER CRYSTALLINE PREPARATIONS ISOLATED IN THE PRESENCE OF VERSENE PRODUCED LIGHT STRAW-COLORED HYDROLYZATES.

PARSE 3

```
 SENTENCE          = INTRØDUCER  CENTER   END-MARK
                                 18       .

18. C1 ASSERTIØN   =  *   SUBJECT           *  VERB          *  ØBJECT   R-V    *
                          1 PRESENCE 5         MAY HAVE         8                17

 1. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                     THE

 5. C20 P N        = L-P   P     N
                           ØF    2 METALS 4

 8. C131           = VEN              *  ØBJECT               R-V
                     CATALYZED           6 DESTRUCTIØN 7

17. C161           = L-CS   CS1      C1 ASSERTIØN
                            SINCE    16

 2. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                                                  TRACE

 4. C20 P N        = L-P   P     N
                           IN    3 PREPARATIØN

 6. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                     THE

 7. C20 P N        = L-P   P     N
                           ØF    TRYPTØPHAN

16. C1 ASSERTIØN   =  *   SUBJECT              *  VERB            *  ØBJECT               R-V    *
                     9    10 PREPARATIØNS         ISØLATED 14        15 HYDRØLYZATES

 3. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE      TYPE NS   NØUN
                     THE                    CRYSTALLINE

 9. B16            = ADVERB
                     LATER

10. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE      TYPE NS   NØUN
                                            CRYSTALLINE

14. C20 P N        = L-P   P     N
                           IN    11 PRESENCE 13

15. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE               TYPE NS   NØUN
                                            LIGHT STRAW-CØLØRED

11. L-N            = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                     THE

13. C20 P N        = L-P   P     N
                           ØF    VERSENE 12

12. C132 PASSIVE   = VEN            *  PASSIVE-Ø  R-V   *
                     PRØDUCED
```

FEB 17 1965

SENTENCE 15. HUMIN FORMATION DID NOT HAMPER THE HANDLING OF THE SAMPLE AND PROBABLY DID NOT AFFECT THE RESULTS.

PARSE 1

```
    SENTENCE           = INTRØDUCER   CENTER   M1          END-MARK
                                      7        AND 8 13    .

 7. C1 ASSERTIØN       =  *    SUBJECT         *  VERB      *  ØBJECT  R-V    *
                               1 FØRMATIØN        DID 2        6

 8. B16                = ADVERB
                         PRØBABLY

13.  Q1 CONJUNCT       = CENTER
                         12

 1. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                                                                  HUMIN

 2. B16                = ADVERB
                         NØT

 6. C136               = V          *  ØBJECT  R-V
                         HAMPER        5

12. C1 ASSERTIØN       =  *   SUBJECT                  *  VERB     *  ØBJECT  R-V    *
                              ( HUMIN FØRMATIØN )         DID 9       11

 5. C83 VING STG       = L-N  VING       XXX          R-V    *
                         3    HANDLING   ØF 4 SAMPLE

 9. B16                = ADVERB
                         NØT

11. C136               = V          *  ØBJECT       R-V
                         AFFECT        10 RESULTS

 3. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                         THE

 4. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                         THE

10. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NØUN
                         THE
```

FEB 17 1965

SENTENCE 15. HUMIN FORMATION DID NOT HAMPER THE HANDLING OF THE SAMPLE AND PROBABLY DID NOT AFFECT THE RESULTS.

PARSE 2 (REDUNDANT)

```
    SENTENCE          = INTRØDUCER   CENTER   M1        END-MARK       .
                                     7        AND 13    .

 7. C1 ASSERTIØN      =  *  SUBJECT           *  VERB      *  ØBJECT  R-V    *
                            1 FØRMATIØN          DID 2        6

13.  Q1 CONJUNCT      = CENTER
                        12

 1. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                                                                     HUMIN

 2. B16               = ADVERB
                        NØT

 6. C136              = V            *  ØBJECT   R-V
                        HAMPER          5

12. C1 ASSERTIØN      =  *  SUBJECT                       *  VERB     *  ØBJECT   R-V    *
                        8   ( HUMIN FØRMATIØN )              DID 9       11

 5. C83 VING STG      = L-N  VING        XXX            R-V     *
                        3    HANDLING    ØF 4 SAMPLE

 8. B16               = ADVERB
                        PRØBABLY

 9. B16               = ADVERB
                        NØT

11. C136              = V            *  ØBJECT        R-V
                        AFFECT          10 RESULTS

 3. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                        THE

 4. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                        THE

10. L-N               = ARTICLE   QUANTIFIER   ADJECTIVE   TYPE NS   NØUN
                        THE
```

Scientific American, January 1962, vol. 206, no. 1.

Aftereffects in Perception

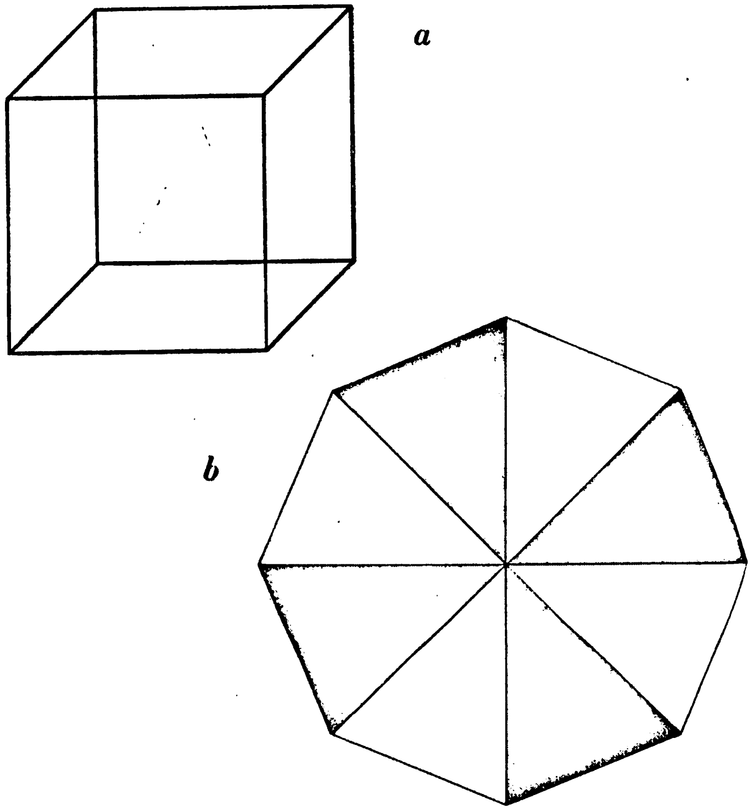
Certain optical illusions involve the reversal of a geometrical figure on prolonged viewing. These aftereffects and similar ones appear to be due to a special electrical phenomenon in the brain

by W. C. H. Prentice

Human experience and human behavior are accessible to observation by everyone. The psychologist tries to bring them under systematic study. What he perceives, however, anyone can perceive; for his task he requires no microscope or electronic gear. A genuine discovery in this field—a wholly new item of experience or a fact about behavior previously unknown—is accordingly rare. Once in a while, nonetheless, such a discovery is made. Within the past few years some simple observations have uncovered new and disturbing facts about human sensory experience, particularly visual experience. Comprehension of these facts is bringing about a radical revision in the prevailing concept of the nature of that experience.

The work that led to this development began with experiments involving familiar optical illusions. One is the illusion associated with the schematic diagram of a cube, in which the lines are drawn to represent the edges of the six faces of the cube [*see "a" in illustration at right*]. If one looks at such a diagram for any length of time, the cube will periodically shift its apparent orientation in space. At first the lower face may seem to be projecting toward the observer. Then the upper face will. This shifting continues for as long as one looks
14 at the figure. Another such illusion is presented by an octagon in which equal sections are alternately dark and light
15 [*see "b" in illustration at right*]. This figure also changes its organization with
16 prolonged viewing. If it appears at first to show a dark cross on a light ground, it will in time shift and appear to show
17 a light cross on a dark ground. These reversals are in large part involuntary. The observer cannot maintain the cube or the octagon in either of their configurations indefinitely. Sooner or later the figures will reverse.

The peculiar behavior of these figures is now known to be an instance of the general class of "figural aftereffects." As the reversibility of the figures suggests, something connected with the initial way of seeing the figure becomes fatigued or satiated, permitting the competing configuration to take over for a while until it too weakens and gives way to the configuration originally perceived. These reactions, which were fir demonstrated by the classical method of psychology, have been correlated wit electrochemical changes in the brai the study of which does involve delicat instruments. The instruments show th a sensory stimulus produces a curre flow through the area of the cerebral cor tex to which the stimulus is relayed an that the current, by satiating and t



PERCEPTUAL ANOMALIES associated with figural aftereffects are shown here. The cub (*a*) and the cross (*b*) periodically reverse with prolonged viewing. Reversibility is a speci case of a general perceptual process, related to the self-limiting direct-current flow in th brain caused by sensory stimulation and to its effects on the future reactivity of brain tissu

AUG 5 1965

```
  PARSE NO. 1

 SENTENCE 14.  ANOTHER SUCH ILLUSION IS PRESENTED BY AN OCTAGON IN WHICH
 EQUAL SECTIONS ARE ALTERNATELY DARK AND LIGHT.


  SENTENCE          = INTRODUCER  CENTER  END-MARK
                                  11       .

11. C1 ASSERTION    =  *  SUBJECT        *  VERB    *  OBJECT  R-V    *
                          1 ILLUSION        IS         10

 1. L-N             = ARTICLE   QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                      ANCTHER   SUCH

10. C132 PASSIVE    = *  VEN             *  PASSIVE-O  R-V    *
                         PRESENTED 9

 9. C20 P N         = L-P  P     N
                           BY    2 OCTAGON 8

 2. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                      AN

 8. C88WH STG       = P     XXX     C1 ASSERTION  M1
                      IN    WHICH   5             AND 7

 5. C1 ASSERTION    =  *  SUBJECT        *  VERB    *  OBJECT  R-V    *
                          3 SECTIONS        ARE   4    DARK

 7. Q1 CONJUNCT     = C1 ASSERTION
                      6

 3. L-N             = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                                           EQUAL

 4. B16             = ADVERB
                      ALTERNATELY

 6. C1 ASSERTION    =  *  SUBJECT               *  VERB      *  OBJECT  R-V   *
                          ( EQUAL SECTIONS )       ( ARE )      LIGHT
```

```
    PARSE NO. 2                                                    AUG 5  1965

   SENTENCE 14.  ANOTHER SUCH ILLUSION IS PRESENTED BY AN OCTAGON IN WHICH
   EQUAL SECTIONS ARE ALTERNATELY DARK AND LIGHT.


   SENTENCE          = INTRODUCER  CENTER   END-MARK
                                   1C       .

10. C1 ASSERTION     =  *   SUBJECT        *   VERB    *   OBJECT   R-V    *
                            1 ILLUSION         IS          9

 1. L-N              = ARTICLE   QUANTIFIER  ADJECTIVE   TYPE NS   NOUN
                       ANCTHER   SUCH

 9. C132 PASSIVE     = *   VEN             *   PASSIVE-O  R-V    *
                           PRESENTED 8

 8. C20 P N          = L-P  P    N               M1
                            BY   2 OCTAGON 6     AND 7

 2. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE   TYPE NS   NOUN
                       AN

 6. C8BWH STG        = P    XXX      C1 ASSERTION
                       IN   WHICH    5

 7. Q1 CONJUNCT      = N
                       LIGHT

 5. C1 ASSERTION     =  *   SUBJECT        *   VERB    *   OBJECT   R-V    *
                            3 SECTIONS         ARE   4     DARK

 3. L-N              = ARTICLE  QUANTIFIER  ADJECTIVE   TYPE NS   NOUN
                                            EQUAL

 4. B16              = ADVERB
                       ALTERNATELY
   PARSE NO. 3                                                     AUG 5  1965

     SENTENCE 14.  ANOTHER SUCH ILLUSION IS PRESENTED BY AN OCTAGON IN WHICH
     EQUAL SECTIONS ARE ALTERNATELY DARK AND LIGHT.


     SENTENCE          = INTRODUCER  CENTER  M1       END-MARK
                                     9       AND 11   .


  9. C1 ASSERTION      =  *  SUBJECT        *  VERB   *  OBJECT  R-V   *
                             1 ILLUSION        IS        8

 11. Q1 CONJUNCT       = CENTER
                         10

  1. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                         ANOTHER  SUCH

  8. C132 PASSIVE      = *  VEN             *  PASSIVE-O  R-V  *
                            PRESENTED 7

 10. C1 ASSERTION      =  *  SUBJECT                          *  VERB     *  OBJECT  R-V
                             ( ANOTHER SUCH ILLUSION )           ( IS )      LIGHT

  7. C20 P N           = L-P  P    N
                              BY   2 OCTAGON 6

  2. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                         AN

  6. C88WH STG         = P    XXX     C1 ASSERTION
                         IN   WHICH   5

  5. C1 ASSERTION      =  *  SUBJECT        *  VERB   *  OBJECT  R-V   *
                             3 SECTIONS        ARE  4    DARK

  3. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                                              EQUAL

  4. B16               = ADVERB
                         ALTERNATELY


NO MORE PARSES.
```

DEC 17 1965

```
SENTENCE 15. THIS FIGURE ALSO CHANGES ITS ORGANIZATION WITH PROLONGED VIEWING .
     PARSE     01
 1. SENTENCE          = INTRODUCER CENTER END MARK
                                    2.     .


 2. C1 ASSERTION      =   * SUBJECT        *    MAIN VERB   * OBJECT            R-V   *
                             3. FIGURE   ALSO   CHANGES       4. ORGANIZATION        5.


 3.  L-A OF N         = ARTICLE QUANTIFIER ADJECTIVE TYPE NS NOUN
                        THIS


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


 5. C164 CS4 SN       = L-CS CS4    VING-STGS
                             WITH    6.


 6. C83 VING STG      =  L-A OF N VING       OF-OBJ R-V    *
                         7.       VIEWING   (( ))


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




      NO MORE PARSES
```

```
PARSE NO. 1                                                    AUG 1 2 1965

SENTENCE 16.  IF IT APPEARS AT FIRST TO SHOW A DARK CROSS ON A LIGHT
GROUND, IT WILL IN TIME SHIFT AND APPEAR TO SHOW A LIGHT CROSS.

SENTENCE           = INTRODUCER  CENTER  END-MARK
                                 12      .

12. C1 ASSERTION   =  *   SUBJECT   *   VERB                        *   OBJECT   R-V   *
                      7   IT            WILL 8 SHIFT AND 9                              11

 7. C161           = L-CS  CS1  C1 ASSERTION
                           IF   6

 8. C20 P N        = L-P  P    N
                          IN   TIME

 9. Q1 CONJUNCT    = V
                     APPEAR

11. C130TO V+OBJ   = L-V  TO   V        *  OBJECT      R-V   *
                          TO   SHOW        10 CROSS

 6. C1 ASSERTION   =  *  SUBJECT   *   VERB        *  OBJECT  R-V   *
                         IT            APPEARS  1     5

10. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                     A                    LIGHT

 1. C19            = P    ADJECTIVE
                     AT   FIRST

 5. C130TO V+OBJ   = L-V  TO   V        *  OBJECT      R-V   *
                          TO   SHOW        2 CROSS 4

 2. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                     A                    DARK

 4. C20 P N        = L-P  P    N
                          ON   3 GROUND ,

 3. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                     A                    LIGHT
```

```
    PARSE NO. 2

    SENTENCE 16.  IF IT APPEARS AT FIRST TO SHOW A DARK CROSS ON A LIGHT
    GROUND, IT WILL IN TIME SHIFT AND APPEAR TO SHOW A LIGHT CROSS.


    SENTENCE       = INTRODUCER  CENTER  END-MARK
                                 12      .

12. C1 ASSERTION   =   * SUBJECT    * VERB           M1        * OBJECT  R-V   *
                      7  IT           WILL 8 SHIFT   AND 9                    11

 7. C161           = L-CS  CS1  C1 ASSERTION
                           IF   6

 8. C20 P N        = L-P  P    N
                          IN   TIME

 9. Q1 CONJUNCT    = SUBJECT   *  VERB
                     ( IT )       ( WILL ) APPEAR

11. C130TO V+OBJ   = L-V  TO   V      *  OBJECT     R-V    *
                          TO   SHOW      10 CROSS

 6. C1 ASSERTION   =   * SUBJECT    * VERB        * OBJECT  R-V    *
                         IT           APPEARS   1   5

10. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                     A                    LIGHT

 1. C19            = P    ADJECTIVE
                     AT   FIRST

 5. C130TO V+OBJ   = L-V  TO   V      *  OBJECT      R-V    *
                          TO   SHOW      2 CROSS 4

 2. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                     A                    DARK

 4. C20 P N        = L-P  P    N
                          ON   3 GROUND ,

 3. L-N            = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                     A                    LIGHT
```

AUG 12 1965

```
PARSE NO. 3                                                    AUG 1 2 1985

SENTENCE 16.  IF IT APPEARS AT FIRST TO SHOW A DARK CROSS ON A LIGHT
GROUND, IT WILL IN TIME SHIFT AND APPEAR TO SHOW A LIGHT CROSS.


SENTENCE              = INTRODUCER  CENTER  M1        END-MARK
                                    9       AND 13    .


 9. C1 ASSERTION      =  *  SUBJECT   *  VERB                *  OBJECT  R-V   *
                         7  IT           WILL 8 SHIFT

13. Q1 CONJUNCT       = CENTER
                        12

 7. C161              = L-CS  CS1  C1 ASSERTION
                              IF   6

 8. C20 P N           = L-P  P    N
                             IN   TIME

12. C1 ASSERTION      =  *  SUBJECT   *  VERB                  *  OBJECT  R-V
                            ( IT )       ( WILL ) APPEAR          11

 6. C1 ASSERTION      =  *  SUBJECT   *  VERB          *  OBJECT  R-V   *
                            IT           APPEARS  1       5

11. C130TO V+OBJ      = L-V  TO   V       *  OBJECT     R-V    *
                             TO   SHOW       10 CROSS

 1. C19               = P    ADJECTIVE
                        AT   FIRST

 5. C130TO V+OBJ      = L-V  TO   V       *  OBJECT     R-V    *
                             TO   SHOW       2 CROSS 4

10. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                        A                    LIGHT

 2. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                        A                    DARK

 4. C20 P N           = L-P  P    N
                             ON   3 GROUND ,

 3. L-N               = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                        A                    LIGHT
```

```
    PARSE NO. 1        JUL 30 1965

    SENTENCE 17.  THESE REVERSALS ARE IN LARGE PART INVOLUNTARY.


    SENTENCE            = INTRODUCER  CENTER  END-MARK
                                      4       .

  4. C1 ASSERTION       =  *  SUBJECT          *  VERB   *  OBJECT          R-V  *
                              1 REVERSALS         ARE   3   INVOLUNTARY

  1. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                          THESE

  3. C20 P N            = L-P  P    N
                               IN   2 PART

  2. L-N                = ARTICLE  QUANTIFIER  ADJECTIVE  TYPE NS  NOUN
                                               LARGE





NO MORE PARSES.
```

04 FERROUS EXTRACTION AND REFINING

M04--24522. THE EFFECT OF CALCIUM CARBONATE ON THE REDUCIBILITY OF IRON-OXIDE AGGLOMERATES. P.K. Strang way 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)

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

```
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
```

```
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
```

```
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
```

```
SENTENCE  4. AT LOW REDUCTION TEMPERATURES , THE EFFECT WAS MORE PRONOUNCED
            AS THE CALCIUM CARBONATE CONTENT INCREASED .
     PARSE     02
 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. C160 CSO O-BE   =      AS  10. CONTENT 11.

 7.  L-A OF N       =                     LOW               REDUCTION

10.  L-A OF N       = THE                                   CALCIUM CARBONATE

11. C132 VEN O-PASS = INCREASED
NO MORE PARSES
```

```
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
```

Figure 1

FEB 10 1966

```
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
```

```
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
```

Report on the String Analysis Programs

IV

The FAP String Analysis Program
Carol Raze

TABLE OF CONTENTS

	Page
INTRODUCTION	1
I - Word Dictionary	2
II - Grammar Strings	2
III - The Tree	3
IV - Analyzing a Sentence	6
A. Building the Parse Tree	6
B. Inserting or Specifying a Grammar Definition	11
C. The Restrictions	17
[Alphabetic list of routines - p. 20]	

TABLE OF FIGURES

1. Machine Representation of a String S	28
2. The Node NS	30
3. Flow Chart of PARSER	32
4. The Grammar Link LT_k	35

25

APPENDIX - The correspondences between the restriction routines of the FAP string analysis program and those of the IPL string analysis program. 36

The FAP String Analysis Program

Introduction

The FAP system which grammatically analyzes sentences consists of three separate programs:

a) the program which generates the word dictionary tape.

b) the program which generates the grammar dictionary tape.

c) the program which analyzes sentences using three inputs - the word dictionary, the grammar dictionary and the input sentence.

Each program is written in FAP for the 7094 and should be run under the FORTRAN II System. The program occupies 10,000[1] core locations, the grammar 8,000. For a typical sentence the word entries occupy 300 locations and the analysis tree 1,000. The upper limit for the length of a sentence is 127 words owing to the node structure (see figure 2). It takes approximately 1 second to obtain the first parse of a sentence and approximately 5 seconds to obtain all possible parses.

It is important to understand the structure of two of the inputs (grammar and word dictionary) to the parsing program and the tree constructed by the program to represent the analysis of a particular sentence. The grammar is composed of strings and restrictions. A grammar string consists of definitions of the grammatical entity which the string represents; a restriction consists of tests to determine whether or not a particular definition of the string is valid for the sentence being analyzed. The word dictionary supplies a category list for each word in the dictionary. This list consists of all the categories of the word: e.g. noun, verb, etc.

26

[1] All figures relating to core locations are approximate.

The sentence analyzer parses a sentence by constructing a tree (from the string definitions) whose terminal nodes correspond to the categories of the successive words of the sentence.

I Word Dictionary

The words are arranged in the dictionary in alphabetical order, each word having a record independent of the others. A word W has a set of category assignments $\underline{C_1 \text{ or } C_2 \text{ or } \ldots \text{ or } C_n}$, where each C_i corresponds to a possible part of speech. C_i is the same as the name of an atomic string in the grammar. The C_i's form a category list C. Each category may have a set of subcategories. Therefore C consists of $\underline{L_1\ C_1 \text{ or } L_2\ C_2 \text{ or } \ldots \text{ or } L_n\ C_n}$ where L_i is $\underline{C_{i1} \text{ and } C_{i2} \text{ and } \ldots \text{ and } C_{im}}$. L_i is a set of properties of W for the C_i category.

II Grammar Strings[1]

A grammar string S is defined to be $\underline{S_1 \text{ or } S_2 \text{ or } \ldots \text{ or } S_n}$. The S_i's are called the options of S. The S_i's are defined to be $\underline{S_{i1} \text{ and } S_{i2} \text{ and } \ldots S_{im}}$. The S_{ij}'s are the elements of the i^{th} option and are themselves grammar strings like S. Therefore we have a system of strings composed of other strings. However, this system is not infinite. The process ends with atomic strings. An atomic string is a symbol which is not composed of any other strings; it corresponds to a word category whereas the non-atomic string represents a broader grammatical entity.

27

[1] In string theory of language structure a string is a sequence of elements $\underline{E_1 \text{ and } E_2 \text{ and } \ldots E_m}$. This corresponds to a special case of "string" as used in this article, the case n=1. For example the linguistic string $S=\underline{N\ tV\ N}$ corresponds to the program string $S=S_1$, where S_1 consists of the elements $\underline{N}$, $\underline{tV}$, $\underline{N}$.

To allow for a more refined grammar, restrictions[1] were added to the options of a string. The complete definition of S is actually $R_1\ S_1$ or $R_2\ S_2$ or ... or $R_n\ S_n$, where R_i is a series of tests to be performed upon the sentence or tree. If R_i fails then S_i is not a proper choice for S.

The strings are represented in the machine as lists[2]. The first word in the list of a string S is the head of S which contains its code name S and certain properties of S. The second word points to $R_1\ S_1$, the first option of S and its restriction. In general the (n+1)-st word of S points to $R_n\ S_n$, the n^{th} option and its restriction. The options are also lists. The list of each option S_i is similar to the list of a string except that it has no head. The first word of S_i points to S_{i1}, the first element of S_i. In general the n^{th} word of S_i points to S_{in}, the n^{th} element of S_i. There is a slight difference in atomic strings. The head of the atomic string has an atomic signal and the string is composed of only a head. See figure (1) for a detailed machine representation of a string.

III The Tree

The tree is constructed by the analyzer program from the grammar strings. It is actually a record of the options and their elements which the program has chosen from the definitions of the strings.

Each unit of the tree is called a node and corresponds to an element of an option. The node N takes up two machine words. N consists mainly of 28 pointers an "UP" or "LEFT" pointer, a "DOWN" pointer and a "RIGHT" which point respectively to the node above or left of N, to the node below N

[1] The restrictions will be discussed in detail in part I V-C of this article.

[2] Several concepts used in this structure were taken from the NU-SPEAK list processing language developed at the Courant Institute of Mathematical Sciences. All lists used by the program are sequential one dimensional arrays. Each element in the list takes up one machine word.

and to the node to the right of N. The absence of any of the above pointers, for example the "RIGHT" pointer would mean there is no node to the right of N. N also has a "GRAMMAR" pointer whose function will be explained later.

Suppose the program has just attached a node NS corresponding to a string S and must now look at the definition of S and choose an option from it. Suppose the option S_i (which consists of elements S_{i1}, S_{i2},...,S_{im}) is the correct choice for S. Nodes NS_{i1}, NS_{i2},...,NS_{im} will be attached to the tree in the following manner:

1) NS will have a "DOWN" pointer to NS_{i1}.

2) NS_{i1} will have an "UP" pointer to NS and a "RIGHT" pointer to NS_{i2}.

3) NS_{i2} will have a "LEFT" pointer to NS_{i1} and a "RIGHT" pointer to NS_{i3}. NS_{i3} to NS_{im-1} will be similar to NS_{i2}.

4) NS_{im} will have a "LEFT" pointer to NS_{im-1} but no "RIGHT" pointer.

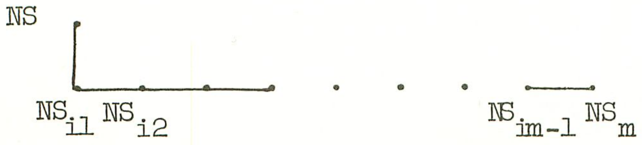
The above position pointers are necessary to connect the various nodes to the tree.

5) Each node will have a "GRAMMAR" pointer to set up the correspondence between NS_{ij} and:

a) S_{ij} , i.e.-the name of the corresponding string

b) its place in the definition of S ; i.e. it occupies the ij^{th} position in the definition of the parent node S.

If we draw the tree starting from S it will look like:



The part of the tree under NS is called the substructure of NS. NS through NS_{im} are all one level below NS; hence NS is their parent (node). NS_{il}, however, is the only node directly below NS. The node structure via the position and grammar pointers shows the particular option of S chosen during the analysis. The context in which S was chosen can be ascertained by looking at the parent node of S. If in the course of building the tree it were found that the particular option used for S is incorrect, the tree would point (via the grammar pointer) to the place in the grammar where the choice was made. This would enable another option for S to be chosen and a different substructure for NS to be constructed.

Since each NS_{ij} corresponds to a string, it will in general have a substructure similar to that of NS shown above. However, if S_{ij} is atomic it cannot have such a substructure since it is a symbol and does not consist of strings. It corresponds to a part of speech, and the current word W must have a category matching S_{ij}. If W does, then NS_{ij} is complete, and it corresponds to the analysis of W. If there is no match the choice was incorrect and a different substructure would have to be built for S, the parent node. When the tree is complete for a sentence it means all the branches are complete and all the words of a sentence correspond to some atomic node of the tree via the matching process. Thus the tree represents a parse of the sentence.

See figure 2 for a detailed machine representation of a node.

IV Analyzing a Sentence

A: Building the Parsing Tree

The parsing program "PARSER" is a subroutine called after the following initialization: (1) the grammar is read in and the initial string is obtained from it; (2) the sentence is read in and the record of each word is extracted from the word dictionary; (3) a sentence list is formed in such a way that the ith word of the sentence list contains the address of the category list of the ith word in the sentence.

PARSER sets up a node NS (the first in the tree) for the initial string S. It will be the only node with no "UP" or "LEFT" pointer.

In what follows, the following symbols are used:

S = string being defined

NS = node corresponding to S

S_{ij} = an element in the definition of S

W = current word being analyzed (initially W is the first word of the sentence)

The logic of PARSER is as follows (cf. Fig. 3):

1 - Constructing the Tree - The first option of the current string S is picked up. NS is the current node. In this phase the program will
31 try to build the substructure of NS.

a) If the current option i has a restriction on it, it must be executed. Certain parts of the restriction, however, may be non-executable. Those parts are noticed at this point but are not executed. They must be retrieved and executed during another phase of the program.

If a restriction fails, go to 1-e and try another option of S. If the restriction succeeds or is nonexistent pick up the first word of the option. That word points to another string S_{i1}. Call it the present string. Go to 1-b.

b) The present string is S_{ij}. If S_{ij} is atomic go to 1-c to see whether the current word matches. Otherwise go to 1-d.

c) The present string S_{ij} is atomic and about to be attached to the tree. At this point the category list in the dictionary entry of the current word W must be searched. One of the categories must be S_{ij},[1] the present string, or else the analysis is incorrect. If there is a match, S_{ij} is acceptable and W has been analyzed.[2] Go to 1-a to set up a node for S_{ij}. If there is no match then

1 - If the present string is the first element in the option (j=1) go to 1-e to try another option.

2 - If the present string is not the first element in the option (j>1) go to 3) to change the substructure of NS.

d) Set up a node NS_{ij} for S_{ij} in the following manner:

1. Place the current word count in the count position of NS_{ij}.
2. If S_{ij} is atomic, place the appropriate signal in NS_{ij} and place the address of subcategory list in the "DOWN"-position of NS_{ij}.

32

[1] Each category has a list of subcategories. If a match is made with S_{ij} the address A of the subcategory list of S_{ij} is saved. When NS_{ij} is created A will be placed in the "DOWN" position of NS_{ij} (this can be done because NS_{ij} is atomic and therefore will never have nodes below it) so that NS_{ij} takes on the properties of W for the S_{ij} category.

[2] There are a few exceptional atomic strings that may be "empty". For these strings no match is attempted for they are always acceptable. However, W has still not been analyzed. Since the grammar strings were defined in a general manner for any sentence this exception makes possible the absence of certain grammatical entities in a particular sentence.

3. If there is a (j+1)-st element in the option S_i, place a "more-elements" signal in NS_{ij}.

4. If j=1, NS_{ij} is the node directly below NS and the first node in that lower level.

   a. Place a "first node in level" signal in NS_{ij}.

   b. Place a pointer to NS_{ij} in the "DOWN" position of NS.

   c. Place a pointer to NS in the "UP" position of NS_{ij}.

   d. Place a pointer to the (i+1)-st word of S in the "GRAMMAR" position of NS_{ij}.

   e. If a wellformedness test was part of the restriction present on the option place a "WELLF" signal in NS_{ij}. This signal means that, after the substructure of NS is complete, the wellformedness test must be executed to see whether or not NS is a well formed node.

5. If $j > 1$, NS_{ij} has a node to the left of it but none directly above it - it is not the first node on the lower level.

   a. Place a pointer to NS_{ij} in the "RIGHT" position of NS_{ij-1}.

   b. Place a pointer to NS_{ij-1} in the "LEFT" position of NS_{ij}.

   c. Place a pointer to the jth word of S_i in the "GRAMMAR" position of NS_{ij}.

Now S_{ij} becomes the current string and NS_{ij} the current node. If S_{ij} is not atomic S_{ij} becomes S and NS_{ij}, NS. Go to 1) to build the substructure of the new NS. If S_{ij} is atomic, the current word was successfully analyzed and unless S_{ij} was an empty atomic, the next word of the sentence becomes current (i.e., the word count is increased

33

by one). A branch of the tree has been completed; go to 2) to complete the tree.

e) An option of S has failed. If there is a next option, make it the current option and go to 1-a to build another substructure for NS. If there are no more options, then all possibilities for S have failed and S itself has failed. The parent of NS is now re-named NS (i.e., it becomes the node to be constructed) and the former NS is re-named NS_{ij}, (i.e., it becomes an element of the parent). Go to 4) to detach NS_{ij} from the tree.

2 - Completing the Tree - A branch of the substructure of NS has been completed. This part of the program will find the next element of the current option (if it exists) and set it up for completion. Test the "more-elements" signal of NS_{ij}:

a) If the current node NS_{ij} has a "more-elements" signal, pick up the (j+1)-st word of the option S_i. This word points to string $S_{i(j+1)}$ which becomes the current element. Remove the "more-elements" signal from NS_{ij}. Go to 1-b to construct another branch of the tree.

b) If the "more-elements" signal is absent, all the nodes have been attached on the present level. Go to NS_{i1} to see if a "WELLF" signal is present, and continue up to NS, the parent node. If there is no parent node, the highest level in the tree has been reached and the tree is now complete[1] - return with the completed

34

[1] At this point it is always guaranteed that every word of the sentence corresponds to some atomic node in the tree because the grammar includes the atomic strings representing end marks (., ?, etc.) as elements of various options.

parse. If there is a parent and the "WELLF" signal was present, retrieve and execute that part of the restriction. If it fails NS was not constructed properly - go to 3) to try to reconstruct NS. If the test succeeds (or if there were no wellformedness test) NS has been completed. Now NS's parent must be completed: NS becomes NS_{ij} and the parent node of NS becomes NS (as in 1-e). Go to 2) to continue the construction of the tree.

3 - Reconstructing the Tree. NS has a substructure which is not correct either because a wellformedness test on NS has failed, or because NS_{ij} ($j > 1$) has just been detached from the tree (i.e., all options of S_{ij} have failed). The most recently attached atomic node, i.e., the rightmost atomic node in the analysis tree at the moment, must be found so that the last completed substructure of the tree may be changed:

a) Go down one node;

b) Go to the rightmost node on the level. It becomes the current node;

c) If the current node is atomic (call it NS_{ij}) go to 4) and detach it. If the current node is not atomic, return to 3a).

4 - Tearing up the Tree - NS_{ij} has failed to fit the sentence correctly. In this phase it is detached from the tree so that another substructure of NS can be built.

a) Detach NS_{ij} from the tree by removing all the appropriate pointers from the surrounding nodes. NS (the parent of NS_{ij}) becomes the current node. However, if NS_{ij} has no parent, it is the first node in the tree. In that case all possible analyses of the sentence have failed and PARSER returns with "no parses" for this sentence.

35

b) If NS_{ij} was a non-empty atomic the previous word in the sentence becomes the current word.

c) If $j > 1$, go to 3) to try to build another substructure for NS. However, if $j=1$, all possible substructures corresponding to option S_i were tried for NS and failed. Go to 1-e to try another option for S.

B: Inserting or Specifying a Grammar Definition

Sometimes it is desirable to temporarily change the definition of a string (substitution) or to insert an extra element into one of its options at a particular point in a sentence (insertion). The mechanism which makes this possible is the grammar link.

Suppose S is the current string and the S_i's its grammar definitions. Let the set T_k be the options obtained from some other source [usually the word dictionary or restrictions]. The T_k's form a list T similar to S (but with no head) and each T_k is identical structurally to an S_i. When a substitution takes place, S has a new set of definitions only at this particular point in the analysis of the sentence. That is, the program treats S as if it were defined to be $\underline{T_1 \text{ or } T_2 \text{ or } \ldots T_\ell}$, although the grammar definition of S remains $\underline{S_1 \text{ or } S_2 \text{ or } \ldots S_n}$. Similarly, if an insertion takes place, S_i has a new definition $\underline{S_{i1} \text{ and } S_{i2} \text{ and } \ldots \text{ and } T_{kl} \text{ and } \ldots S_{im}}$, where k could be $\underline{1 \text{ or } 2 \text{ or } \ldots \text{ or } \ell}$, only at this particular point in the analysis. T_k has only one element (the element to be inserted) although T itself may have various options. 36

Before the node NT_{kl} corresponding to T_{kl} (for substitution or insertion) is set up, a grammar link LT_k must be created in the following manner:

1) The regular grammar pointer (the one that would have been placed in the node if the substitution or insertion had not occurred) is placed in the "grammar" position of LT_k.

2) The address of the kth word of T is placed in the "specify" position of LT_k.

3) If this is a substitution, a signal is placed in the appropriate position of LT_k.

4) If this is an insertion, a signal is placed in the appropriate position of LT_k.

Now NT_{k1} may be set up. It will be like any node except that the address of LT_k is placed in the "grammar" position of NT_{k1}. Thus the program can easily distinguish between regular nodes and those created by substitution or insertion and also be able to save the necessary correspondences. During a substitution, the nodes corresponding to the remaining elements are similar to regular middle-of-a-level nodes, i.e. for NT_{kp}, $p > 1$, the address of the pth word of T_k would be placed in the grammar position of the node. (Note that p cannot be greater than 1 for an insertion.)

See Fig. (4) for a detailed machine representation of a grammar link.

1 - Special Process (a case of insertion)

The grammar does not contain coordinate conjunctional strings in the main body of strings. If conjunctions (and other "special" words of the language) were accounted for explicitly in the strings, the grammar would attain very large proportions. To avoid this, a "special process" mechanism exists in the program which allows for the insertion of an element in an already defined string of the grammar upon the appearance of a conjunction (or other "special process" word) in the sentence.

Suppose W has just been analyzed and W + 1 has become the current word. If W + 1 has a special process mark M on its category list, the sublist of M is obtained. It is a list T, defined to be $\underline{R_1T_1 \text{ or } R_2T_2 \text{ or } \ldots \text{ or } R_nT_n}$. R_k is a series of tests and T_k is an option of T. If NS_{ij} is the current node, a node NT_{kl} is attached (if R_k permits) to the right of it. Thus when (and if) NS is complete, the nodes $NS_{i1},\ldots,NS_{ij},NT_{kl},\ldots NS_{im}$ appear below it, as if S_i were defined to be $\underline{S_{i1} \text{ and} \ldots \text{and } S_{ij} \text{ and } T_{kl} \text{ and} \ldots S_{im}}$. Each element T_{kl} is the grammar string which contains the definition of the special process word. Thus if W is "and" its T list has one option T_1 and T_1 has one element T_{11} which is the grammar string representing conjunctions. However, words marked special may appear in a sentence in one of their non-special uses. Therefore a word with a special mark M is first treated specially, and if no analyses can be produced the special process marker M must be ignored at this point-NS_{ij}-in the analysis and the regular procedure followed until the analysis of NS is complete.

Suppose W, the current word, has an M on its category list. NS_{ij}

is the current node and it is complete either because it is atomic or because its substructure is complete. The following takes place:

a) Constructing the Special Process Node

Get the sublist T of M. Pick up the first option.

1) If T_k has a restriction on it execute it. If it fails go to a-4 to choose the next option. If there is no restriction, or if the restriction succeeds, go to a-2 to set up a grammar link.

2) Set up a link LT_k in the following manner:

a) Place an insertion signal in LT_k

b) Place the address of the kth word of T in the "specify" position of LT_k.

c) If NS_{ij} has a "more-elements" signal, place the address of the (j+1)-st word of S_i in the "grammar" position of LT_k.

d) If NS_{ij} does not have a "more-elements" signal, place 0 in the "grammar" position of LT_k.

3) Set up a node NT_{kl} to the right of NS_{ij} in the same way as any other node, except that the address of LT_k is stored in the "grammar"position of NT_{kl}. Also, if NS_{ij} has a "more-elements" signal it is removed and placed in NT_{kl}. The substructure of NT_{kl} is constructed as usual.

4) If there is a next option, make it T_k and go to a-1. If there are no more options, all the possibilities for special W have failed at this point in the analysis. Follow the regular procedure at NS_{ij}.

39

b) Detaching a Special Process Node

NT_{kl} is about to be detached.

1) Detach it and free LT_k. Go to a-4 to choose another option.

c) Completing a Special Process Node

NT_{kl} is complete. If it has a "more-elements" signal look at the "grammar" position of LT_k and set up a node to the right of NT_k corresponding to that pointer. If the "more-elements" signal is absent, go to the parent node NS in the usual manner.

2 - Substitution

In certain circumstances it is desirable to substitute the options of a string so that a shortcut through the grammar may be effected. One such case presents itself when the object of a verb is reached. Since no verb can take all of the possible object strings of the grammar, it is very convenient to substitute for the set of all object strings just that subset which occurs with that particular verb. This subset is associated with the verb category of the word.

It is one of the functions of certain parts of the restriction (specify routine) to find whether such a substitution can be effected and if so to supply the substitute set of options. If the specify restriction does this successfully it will point to the substitute list:

40 a) Constructing a Substituted Substructure

Parser has just chosen option S_1 of S. The specify routine in the restriction on the option was successfully executed and it points to a set T of substitute options for S. The first option T_1 is picked up.

1) If T_k has a restriction, execute it. If it fails go to a-4; otherwise go to a-2.

2) Set up a link LT_k in the following manner:

   a) Place the address of S_1 in the "grammar" position of LT_k.

   b) Place the address of T_k in the "specify" position of LT_k.

   c) Place a substitution signal in LT_k.

3) Attach node NT_{k1} below NS in the usual manner except that the address of LT_k is placed in the "grammar" position of NT_{k1}. Follow regular procedure in building the substructure of NT_{k1}.

4) Choose the next option, which becomes T_k, and go to a-1. If there are no more options S has failed. Detach S in the usual manner.

b) Completing a level for the Specified Node

NT_{k1} has just been completed. If there is a "more-elements" signal look at T_{k2} and set up a node NT_{k2} for it. This node will be the same as any middle-of-a-level node, with the grammar pointer to the second word of T_k. Thus only the first node in the level is treated and marked as a substitute.

c) Detaching a Substituted Node

NT_{k1} must be detached.

The option T_k has failed. Detach NT_{k1} and free LT_k. Go to a-4 to choose the next option.

41

C: The Restrictions[1]

A restriction is a series of routines with their arguments which operate in the tree or any of the lists (grammar, word dictionary, sentence lists). The restrictions are part of the grammar and therefore determined by the grammarians. However, the function of the routines in the analyzer program will be described below. By means of these routines the tree or list structure may be examined for different properties, e.g. wellformedness of substructures. A restriction is itself represented in the machine as a list. Each routine in the restriction is executed in order; if any routine in the list fails the restriction fails. When the restriction is encountered, the machine is "looking at" either a node or a word in a list. If a restriction fails, it always returns to its starting point (node or list word); if it succeeds it remains just where the last routine exited. If a routine fails it also returns to its starting point; however, if a routine is successful it either returns to its starting point or leaves the machine "looking at" a different place depending on its function. Some routines must start at nodes and others at list words. Some can differentiate between the two structures and those may start at either place.

The routines will be broken up into groups according to related functions and named below. A more detailed description will follow this summary.

[1] The linguistic restriction is a special case of "restriction" as used in this article. Each restriction of the type already described for the IPL program corresponds to an element of a major routine's argument as follows:

FAP	IPL
WELLF	Wellformedness - R2
DSQLF	Disqualify - R3
SPECF	Specify - R1
OMITF	Omission - R4
CHECF	Atom Check - R7

1. Major Routines - A restriction list composed of these routines is the restriction R_i (defined in part II) found on the option S_i of S.

   a) Nonexecutable Routines - these routines are ignored when the restriction list is executed. They must be retrieved by other routines or other parts of the program and executed by them. These routines mark certain restrictions which must be executed at times other than "option-choosing" time. They are:

      WELLF, FREZF, SUBJR, OLIST, VERBR, PRINT

   b) Executable Routines - these are executed in the normal course of parsing:

      DSQLF, OMITF, SPECF, CHECF

2. Logical Routines - These routines perform logical tests or operations upon other restriction lists:

   TRUE, AND, ORR, IMPLY, CANDO, NOT, COMMN, ORPTH, ITER, EXPNT

3. Climbing Routines - These routines move around the tree or a list:

   a) Tree Routines - UPONE, UPTRN, UPTO, PARAL, LEFT, RIGHT, DOWN1, DOWN, DWNTO, DNTRN, DNRIT

   b) List Routines - NEXTL, PREVL, DWNTO, ATTRB, INTOL, NOTEL

4. Property Routines - These routines test certain properties of the tree, list or sentence:

   a) Tree Routines - OMITD, NOATM, COPY, EMPTY, ISIT, ATTRB

43 b) List Routines - ISIT, ATTRB

   c) Sentence Routines - PARSE, MINWD, WORDL, SENTL, RARE

5. TREE-TO-LIST ROUTINES - These routines use the tree to get to a list:

   FRSTL, LASTL, INTOL, EXEC, ATTRB

6. <u>LIST PRODUCING ROUTINES</u> - These routines work on the tree or a list to produce another list:

   GENER, EDIT, SPCFY

7. <u>REGISTER ROUTINES</u> - These routines use certain "registers" to save and restore nodes or list words:

   STORE, LOOKT

A detailed alphabetical description of the routines will follow. In order to avoid unnecessary repetition the routines will be represented as functions and certain symbols will be used to describe various details. Namely:

1) F(Z) = A routine that doesn't necessarily return to its starting point.

2) T(Z) = A routine that always returns to its starting point.

3) The following variables will describe the type of argument of the routine.

   a) a = a symbol

   b) y = a restriction list

   c) A = a list of symbols $a_1, a_2, \ldots, a_n$

   d) Y = a list of restriction lists $y_1, y_2, \ldots, y_n$

   e) 0 = no argument

   f) X = grammar register (Special location available to the grammarian. It is used in a restriction for storing and retrieving nodes or list words.)

44 4) The subscripts V_1-V_2 attached to F or T indicate where the machine must start (V_1 position) and where it will end (V_2 position).

   a) V = N represents a node

   b) V = L represents a list word

   c) V = 0 means that the routine's functioning is independent of the starting (V1) or stopping (V2) point.

Name of Routine		Type	Operation[1]
AND		$T_{o-o}(Y)$	Test that all y_i's exit +.
ATTRB	1.	$F_{N-L}(O)$	The current node NS must be atomic. Therefore, it corresponds to category S of the word which matches NS. Go to the sublist of category S.
	2.	$F_{L-L}(O)$	Go to the sublist of the current list word.
	3.	$F_{N-L}(A)$	Perform 1, then go down the sublist until an a_i is reached.
	4.	$F_{L-L}(A)$	Perform 2, then go down the list, until an a_i is reached.
CANDO		$T_{o-o}(y)$	Test that y can be executed successfully.
CHECF		$T_{o-o}(A)$	Test whether the current word has an a_i on its category list.
COMMN	1.	$T_{o-o}(A_1A_2)$	Test that a symbol in A_1 matches a symbol in A_2.
	2.	$T_{o-o}(yA_2)$	Test that the following be done: Execute y successfully. If y leads to a node NS form a list A_1 consisting of the symbol S. If y leads to a list set the list equal to A_1. Go to 1.
	3.	$T_{o-o}(A_1y)$	Do step 2. for A_2.
		$T_{o-o}(y_1y_2)$	Do step 2, then 3.

45

[1] If the operation can be performed, the routine is successful and exits +; otherwise the routine fails and exits -. If every routine in a restriction y exits +, then y itself exits +; if any routine in y exits -, y exits -.

Name of Routine	Type	Operation
COPY	$T_{N-N}(0)$	Test whether this node was copied from a saved structure.[1]
DNRIT	$F_{N-N}(0)$	Go to the rightmost node, one level below the current node.
DNTRN	$F_{N-N}(AA')$	Descend to a node Na_i below the current node. Descend in the following manner: (1) Set m = 1; (2) Descend to an Na_i m levels below the current node, scanning from left to right. If there are none, set m = m+1 and go to (2). During the descent, if any node in level m is nontransparent do not go below it, unless it is an Na_i'. Exit-when a further descent is no longer possible, either because there are no more transparent nodes or because the lowest node of the tree has been reached.
DOWN	$F_{N-N}(0)$	Go to the node directly below the current node.
DOWN1	$F_{N-N}(0)$	Go to the node directly below the current node. The node below must have no node to its right (except for special process nodes).
DSQLF	$T_{N-N}((y_1a_1) \ldots (y_na_n))$	Test whether all y_i's can be executed successfully: 1. Set i = 1 2. Empty all grammar registers 3. Execute y_i successfully and return to the starting point. 4. Set i = i+1, Go to 2.

[1] See description of IPL "Recorder" for a discussion of "saving."

Name of Routine		Type	Operation
DWNTO	1.	$F_{N-N}(A)$	Go to a node Na$_i$ below the current node using the same manner of descent as DNTRN except that all nodes are treated as if they were transparent.
	2.	$F_{L-L}(A)$	Go to the place in the present list that has an a$_i$ category.
EDIT		$F_{L-L}(y)$	The machine must be "looking at" the first option of a list L of options. Generate a list L' of options from L in the following manner: 1. Set i = 1 2. Look at the ith word of L (which points to L$_i$, the ith option of L) 3. Test whether y can be executed successfully. If so, put the ith word of L in L'and go to y; if not, go to 4. 4. If there is another option in L set i = i+1 and go to (2); if not, go to 5. 5. Look at L'. It must have at least one option.
EMPTY		$T_{N-N}(O)$	Test whether the current node is empty. I.e. that no node in its substructure corresponds to a word of the sentence.
EXEC	1.	$F_{N-L}(O)$	NS is the current node; NS$_{i1}$ is below it. Look at the (i+1)-st word of the string S.
	2.	$F_{N-O}(RT(y))$	Perform 1, then get the restriction R$_i$ and findthe routine RT on R$_i$. If RT is found, execute y; if RT is not found, find the routine OLIST and execute its argument.

47

Name of Routine		Type	Operation
EXPNT		$F_{o-o}(y)$	Execute y.
FREZF		$F_{o-o}(y)$	This is a nonexecutable routine. It is retrieved and y executed by PARSER when the substructure of a node with the "FREEZE" property is about to be changed. If NS is the node and y is successful, NS is "frozen."
FRSTL		$F_{N-L}(0)$	Go to the place in the sentence list corresponding to the word that was current just before the current node was constructed.
GENER		$F_{N-L}(0)$	The current node must be a special process node. At least one node NS_{ik} must be to the left of it. Generate an option list T, so that $T_1 = \underline{S_{ik}}$ $T_2 = \underline{S_{ik-1} \text{ and } S_{ik}}$ $T_n = \underline{S_{i1} \text{ and } S_{i2} \text{ and } \ldots\ S_{ik}}$ Look at T
IMPLY		$T_{o-o}(y_1 y_2)$	If y_1 exits -, IMPLY exits + ; if y_1 exits +, then IMPLY succeeds only if y_2 exits +.
INTOL	1.	$F_{N-L}(0)$	NS_{ij} is the current node. Look at the jth word of the option S_i.
	2.	$F_{L-L}(0)$	The current list word is pointing to another list. Go to that list.

Name of Routine		Type	Operation
ISIT	1.	$T_{N-N}(A)$	Is the current node an Na_i ?
	2.	$T_{L-L}(A)$	a) If the current list word is an option: is the first element in the option an a_i ? b) If the current list word is an element of an option: is the element an a_i ?
ITER	1.	$F_{o-o}(y)$	Execute y successfully at least once. Then keep executing y until it fails.
	2.	$F_{o-o}(y_1 y_2)$	Execute y_1 successfully in the following manner: a) Execute y_1. If it is successful, ITER exits +. If it is not successful go to b). b) Execute y_2. If it is successful go to a). If it is not, ITER fails.
LASTL		$F_{N-L}(0)$	NS is the current node. Go to the place in the sentence list corresponding to the word that was current when NS was completed.
LEFT		$F_{N-N}(0)$	Go left until the first node which is not a special process node is reached.
LOOKT		$F_{o-o}(X)$	Go to whatever is stored in X. If X is empty, exit -.
MINWD		$T_{o-o}(0)$	Is the current word equal to the MIN-WORD (the closest word to the beginning of the sentence reached during backtracking)?[1]
NEXTL	1.	$F_{L-L}(0)$	Go to the next list word.
	2.	$F_{N-L}(0)$	NS_{ij} is the current node. Go to the (j+1)st word of option S_i.

49

[1] See "Report on String Analysis Programs" for a fuller description.

Name of Routine	Type	Operation
NOATM	$T_{N-N}(O)$	Is the current node non-atomic?
NOT	$T_{O-O}(y)$	If y exits +, NOT exits - ; If y exits -, NOT exits + .
NOTEL	$F_{O-L}(A)$	Look at list A.
OLIST	$F_{O-O}(y)$	This is a non-executable routine. EXEC or the final output program retrieves it and executes its argument y.
OMITD	$T_{N-N}(O)$	Does the present node contain an omission mark (left by OMITF)?
OMITF	$T_{N-N}((y_1a_1)\dots(y_na_n))$	Test whether any y_i can be executed successfully: 1. Set i = 1 2. Empty all grammar registers. 3. Execute y_i. If it is successful leave an "OMITF" mark in the node y_i lead to. If it failed repeat step 2) for the next y_i.
ORR	$T_{O-O}(y)$	The y_i's are executed successively starting with y_1. As soon as any y_i exits +, ORR exits + . If no y_i exits +, ORR exits - .
ORPTH	$F_{O-O}(Y)$	Execution identical to ORR. On completion, ORPTH remains where the successful y_i has brought it.
PARAL	$F_{N-N}(O)$	Go to the node parallel to the current node.

Name of Routine	Type	Operation
PARSE	$T_{o-o}(0)$	Was a parse obtained for this sentence?
PREVL	$F_{L-L}(0)$	Go to the previous list word.
PRINT	$F_{N-O}(y)$	This routine is non-executable. It must be retrieved and its argument y executed by the final output program. If y succeeds, the node is treated as a transparent node in the print-out.
RARE	$T_{o-o}(0)$	Is the RARE SWITCH on? During the first attempt to analyze a sentence the RARE SWITCH is off; if PARSER returns with "no parses", the RARE SWITCH is turned on and PARSER is called again.
RIGHT	$F_{N-N}(0)$	Go right until the first node which is not a special process node is reached.
SENTL	$F_{O-L}(0)$	Go to the place in the sentence list corresponding to the first word of the sentence.
SPCFY	$F_{O-L}(S)$	Set up a list T (of options) composed of one option T_1. T_1 is composed of one element S. Look at T.
SPECF	$F_{N-L}((y_1a_1)\ldots(y_na_n))$	This routine must be on the restriction R_1 on S_1 and it always exits + . It will either find a substitute set of options for S, or leave S unchanged. 1. Set i = 1 2. Empty all grammar registers. 3. Execute y_i. 4. If y_i is successful, the machine is now "looking at" a substitute set

51

Name of Routine	Type	Operation
		of options. Ignore the remaining routines on R_1 and return to PARSER with a "substitution" signal. 5. If y_i is not successful, set $i = i+1$ and go to 2.
STORE	$T_{O-O}(X)$	Store the address of the current node or list word in X.
SUBJR	$F_{O-O}(y)$	This is a non-executable routine. It must be retrieved and y executed by EXEC. y is the path to the subject.
TRUE	$T_{O-O}(O)$	Always exit + .
UPONE	$F_{N-N}(O)$	Go to the parent node of the current node.
UPTO	$F_{N-N}(A)$	Go to an Na_i above the current node.
UPTRN	$F_{N-N}(AA')$	Go to an Na_i above the current node; however, do not go above a non-transparent node unless it is an Na_i'.
VERBR	$F_{O-O}(y)$	This is a non-executable routine. It must be found and y executed by EXEC. y is usually the path to the verb.
WELLF	$T_{N-N}((y_1a_1)\dots(y_na_n))$	This is a non-executable routine. It must be retrieved and y executed by PARSER after NS is complete. Then it is executed in the same manner as DSQLF.
WORDL	$F_{O-L}(O)$	Go to the place in the sentence list corresponding to the current word W.

52

Figure 1-1

Machine Representation of a String S

S　　is defined to be $\underline{S_1 \text{ or } S_2 \text{ or } \ldots S_n}$

S_i　　is defined to be $\underline{S_{i1} \text{ and } S_{i2} \text{ and } \ldots S_{im}}$

S_{ij}　　is a string similar to S

The alphabetical symbols represent fields which are described below.

The List of S

head of S (1st word)	a	b	c	d	e	f	g	h	i	j	k	m	S(Name of String)
2nd word	p	n	Address of restriction of S_1										Address of S_1

....

(n+1)-st word	p	n	Address of restriction on S_n	Address of S_n

The List of S_i

1st word	a	n	Address of S_{i1}
2nd word	p	n	Address of S_{i2}

....

53

m-th word	p	n	Address of S_{im}

Figure 1-2

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
e	Special Empty Atomic
f	Transparent
g	Save
h	Freeze
i	Recursive
j	Min word
k	Special Process
m	~~Conditional Transparency~~ Repetitive property
n	Always 0 - Signal for a non-head
p	Always 1 - Signal for the middle of a list

Figure 2-1

The Node NS_{ij}

a	b	c	d	e	f	g	h
b	b	b	k	b	b	b	l

The ten different fields of a node are:

a: 1 bit - If a certain "OMITF" restriction leads successfully to this node a one (1) will be left in this field.

b: 7 bits - the count field. It contains the word count that was current before NS_{ij} was attached to the tree. The word count of W is the position of W in the sentence. Since b consists of 7 bits, the sentence is limited to 127 words.

c: 1 bit - one (1) for an atomic node.

d: 15 bits - Grammar pointer -

a) if $j = 1$, it is the address of the $(i+1)$-st word of S.

b) if $j > 1$, it is the address of the jth word of S_i.

e: 1 bit - Signal for first node in level (i.e. $j = 1$)

f: 1 bit - "MORE-ELEMENTS" signal indicating that there is a $(j+1)$-st
55 element in the ith option.

g: 1 bit - "WELLF" signal indicating that a wellformedness test must be executed when N, the parent node, is complete.

h: 15 bits. a) if $j = 1$, h is the address of the node above NS_{ij};

b) if $j > 1$, h is the address of the node to the left of NS_{ij}.

Figure 2-2

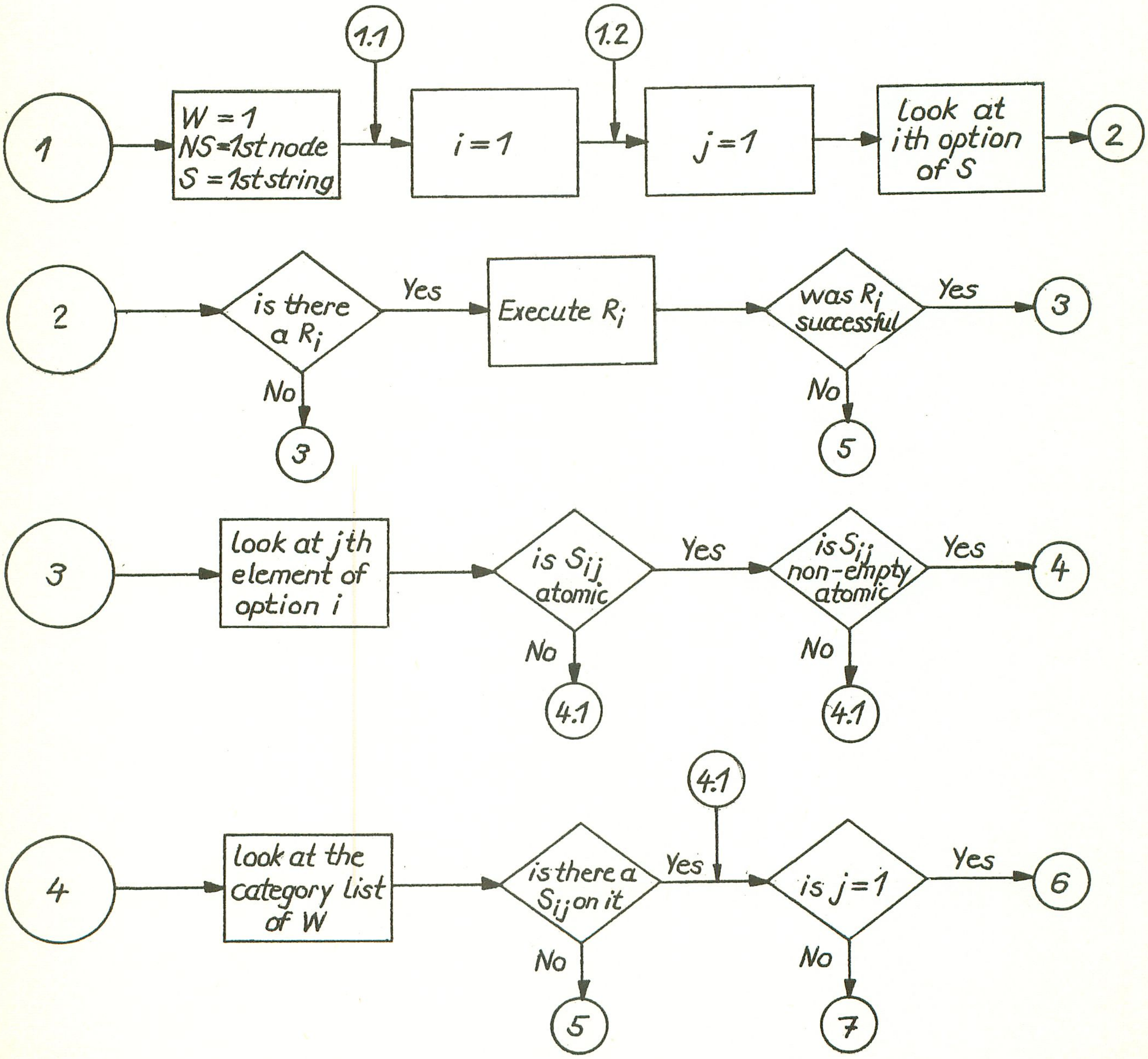
k: 15 bits - Address of node below NS_{ij}. However, if NS_{ij} is atomic, this address is actually that of the subcategory list for the S_{ij} category of the matching word W.

l: 15 bits - Address of node to the right of NS_{ij}.

Figure 3-1

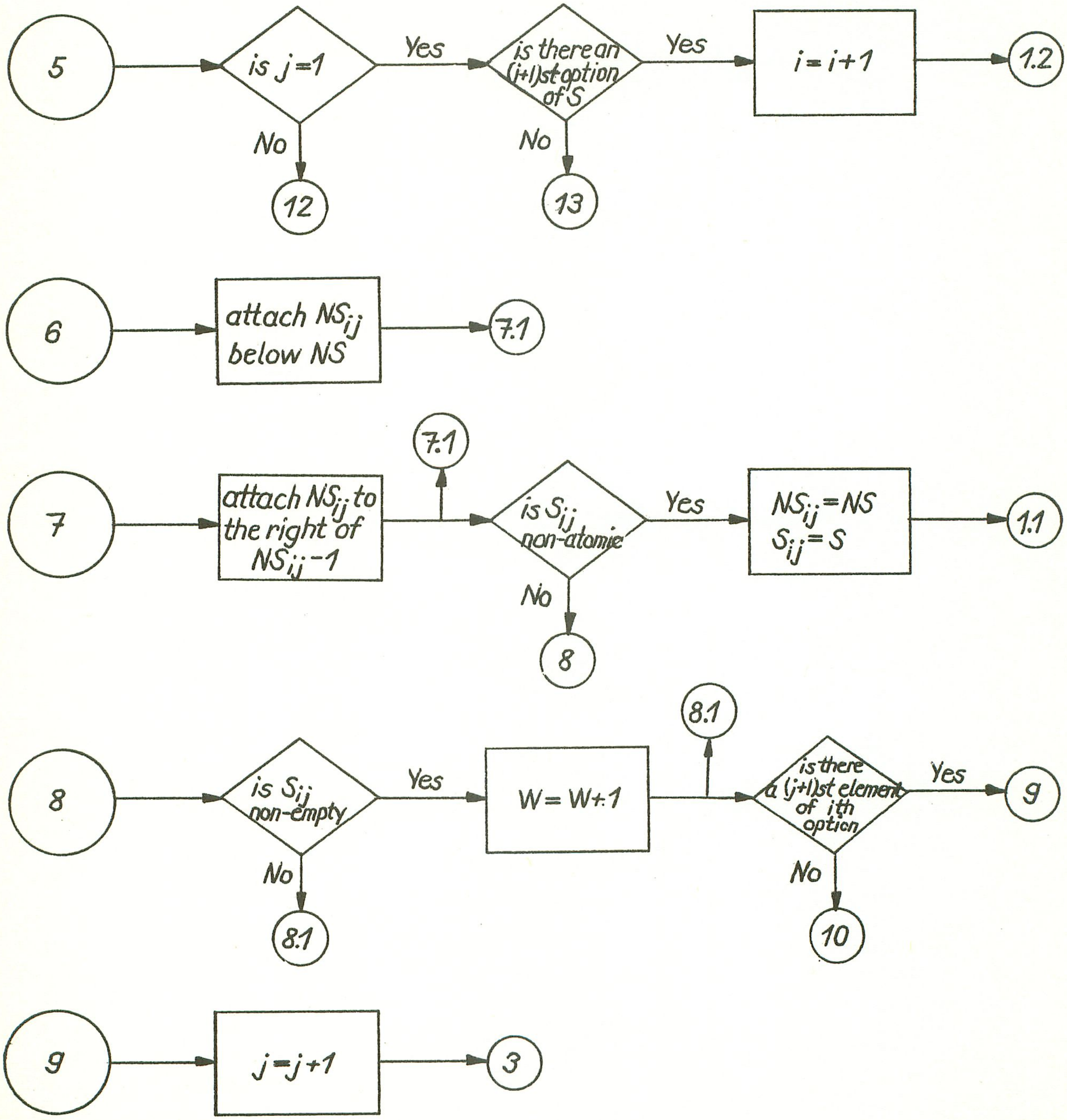
BUILDING THE PARSING TREE

S = string being defined
S_{ij} = jth element of ith option of S
NS = node corresponding to S
W = current word to be analyzed
R_i = restriction on ith option of S



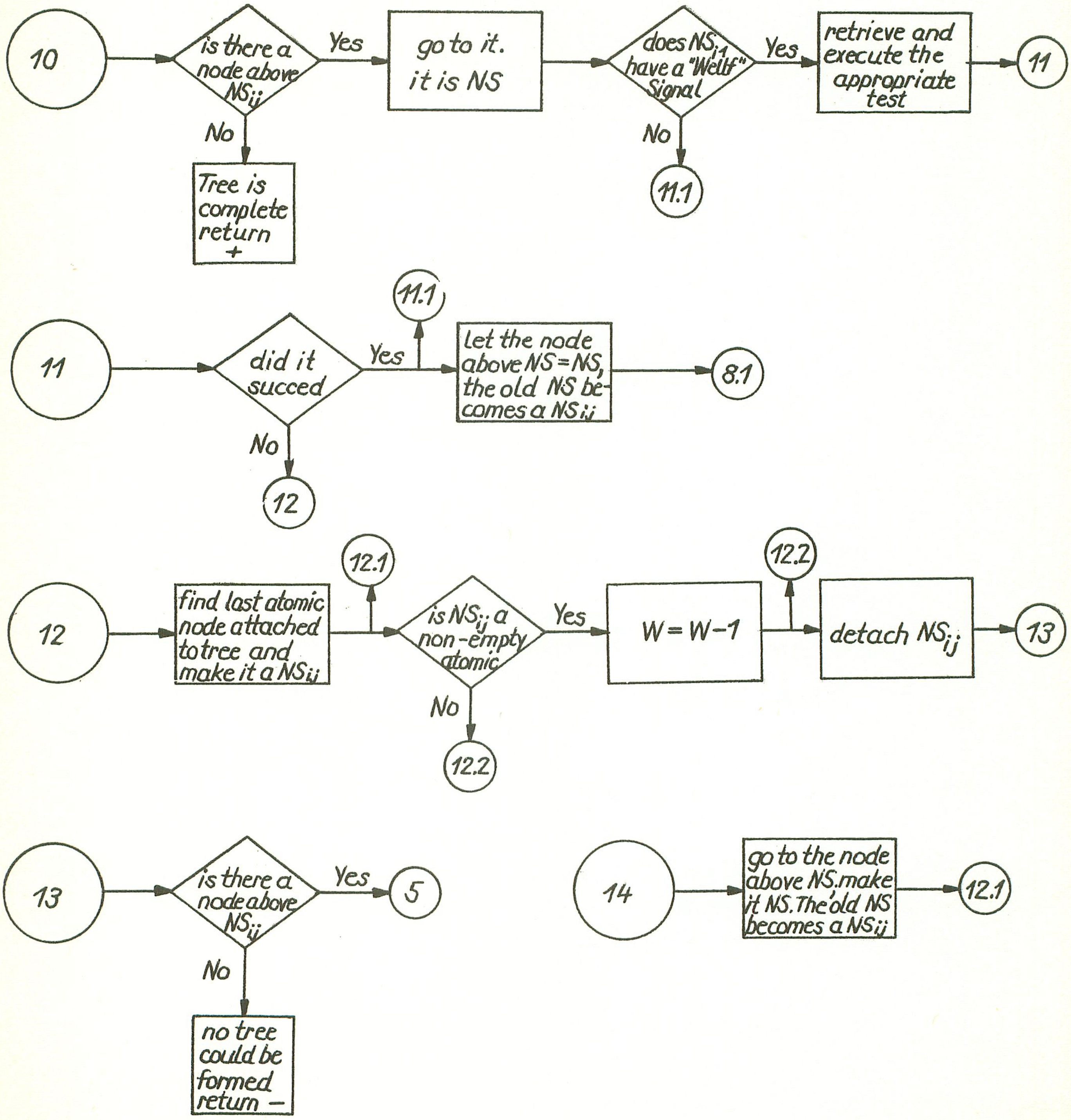
57

Figure 3-2


5
is j=1
Yes
No
12
is there an (j+1)st option of S
Yes
No
13
i=i+1
1.2
6
attach NS_{ij} below NS
7.1
7
attach NS_{ij} to the right of $NS_{ij}-1$
7.1
is S_{ij} non-atomic
Yes
No
8
$NS_{ij}=NS$
$S_{ij}=S$
1.1
8
is S_{ij} non-empty
Yes
No
8.1
W=W+1
8.1
is there a (j+1)st element of ith option
Yes
No
9
10
9
j=j+1
3

58

Figure 3-3


10
is there a node above NS_{ij}
Yes
go to it. it is NS
does NS_{i1} have a "Wellf" Signal
Yes
retrieve and execute the appropriate test
11
No
Tree is complete return +
No
11.1
11
did it succed
Yes
11.1
let the node above NS = NS, the old NS becomes a NS_{ij}
8.1
No
12
12
find last atomic node attached to tree and make it a NS_{ij}
12.1
is NS_{ij} a non-empty atomic
Yes
W = W-1
12.2
detach NS_{ij}
13
No
12.2
13
is there a node above NS_{ij}
Yes
5
No
no tree could be formed return -
14
go to the node above NS. make it NS. The old NS becomes a NS_{ij}
12.1

59

Figure 4

The Grammar Link LT_k

a	b	Address of T_k	c	d	Regular "Grammar" pointer

a - Always 0 (beginning of a list)

b - Always 0 - (Non-head)

c - Signal for insertion

d - Signal for substitution

APPENDIX

The following are the approximate correspondencesbetween the routines of the FAP string analysis program and routines, lists and node attributes of similar function in the IPL string analysis program.

FAP	IPL
AND	/10
ATTRB	---
CANDO	/15
CHECF	R7
COMMN	/17
COPY	*11
DNRIT	/33
DNTRN	/31
DOWN	---
DOWN1	/30
DSQLF	R3
DWNTO1	/32
EDIT	/47
EMPTY	/8
EXEC	/42
EXPNT	---
FREZF	L12
FRSTL	*1
GENER	/46
IMPLY	/12
INT L	/44
ISIT	/7
ITER	/41
LASTL	*6
LEFT	/23

APPENDIX - continued

FAP	IPL
LOOKT	---
MINWD	
NEXTL	---
NOATM	*5
NOT	/16
NOTEL	/45
OLIST	0
OMITD	*7
OMITF	R4
ORR	/11
ORPTH	/40
PARAL	/27
PARSE	*30
PREVL	---
PRINT	---
RARE	*31
RIGHT	/24
SENTL	L3
SPCFY	/45
SPECF	R1
STORE	/43
SUBJR	S
TRUE	/0
UPONÈ	/20
UPTO	/22
UPTRN	/21
VERBR	V
WELLF	R2
WORDL	*10

62