A QUASI-ARITHMETICAL NOTATION FOR SYNTACTIC DESCRIPTION

YEHOSHUA BAR-HILLEL

Massachusetts Institute of Technology

The purpose of this paper* is to present the outlines of a method of syntactic description that is new insofar as it combines methods developed by the Polish logician Kasimir Ajdukiewicz[1] on the one hand and by American structural linguists[2] on the other. Such a combination has apparently not been undertaken before, if only for the reason that Ajdukiewicz's paper appeared in a Polish philosophical journal and has therefore remained unknown to most linguists.

We are not interested here in developing a method which a linguist might use to ARRIVE at an analysis of a linguistic corpus, but only in a new way in which he could PRESENT the results of his investigations. The decisive difference between this method and the others prevailing so far lies in the fact that in addition to a list in which each linguistic element (usually each word) is assigned to one or more categories, only a simple rule of a quasi-arithmetical character need be given to enable us to 'compute' the syntactic character of any given linguistic string (sequence of one or more elements) in its context. The main economy produced by this method lies, therefore, in that it enables us to dispense completely, at least in principle, with special syntactic statements.

This should be of value in those situations in which a completely mechanical procedure is required for discovering the syntactic structure of a given string. Such a situation arises, for instance, in connection with the problem of mechanized translation. It has been shown[3] that a machine could be constructed which would be able to determine the structure of any sentence in the source language, provided that the syntax of this language were presented to the machine in a certain specific form which we may call OPERATIONAL. It may well be that the preparation of the element-category list will be a decisive step toward the construction of an operational syntax whose instructions could be carried out satisfactorily by a digital computer or, for that matter, by a human being operating in a completely mechanical fashion.

Before we proceed to sketch the new method in abstracto, we will discuss one simple example. The English string *Poor John sleeps* would be analyzed, accord-

* This work has been supported in part by the Air Materiel Command, the Signal Corps and the Office of Naval Research, and in part by a grant from the Rockefeller Foundation.

[1] Die syntaktische Konnexität, *Studia philosophica* 1.1–27 (1935). An English mimeographed translation, under the title Syntactic Connection, appeared at the College of the University of Chicago, March 1951.

[2] As presented, for instance, by Zellig S. Harris, *Methods in structural linguistics* (Chicago, 1951), or by Charles C. Fries, *The structure of English* (New York, 1952).

[3] Yehoshua Bar-Hillel, The present state of research on mechanical translation, to appear in *American documentation*.

ing to one method[4] recently described, in the following way: *poor* is an A (for adjective), *John* is an N (for noun), *sleep* is a V (for verb), *-s* is a Vv (for morpheme added to a verb to form a verbal phrase), where all these assignments hold at least for the given context. Then, by invoking the syntactic statements, $A^\frown N \rightarrow N$ (where the arch designates concatenation and the arrow stands for *yields*) and $V^\frown Vv \rightarrow V$, *Poor John* would be assigned to N, *sleeps* to V, hence finally *Poor John sleeps* to $N^\frown V$ which is the most frequent form of English sentences.

According to the notation to be proposed and explained in this paper, *John* will belong to the category n, *poor* to $\frac{n}{[n]}$, *sleeps* to $\frac{s}{(n)}$, where n is to be interpreted, approximately, as the category of name-like strings, $\frac{n}{[n]}$ as the category of those strings that with an n to their right form a string belonging to the same category n, and $\frac{s}{(n)}$ as the category of those strings that with an n to their left form a string belonging to the category of sentences. That the string *Poor John sleeps* is a sentence can now be tested mechanically, without recourse to any syntactic statements, by using something like ordinary arithmetical multiplication of fractions on the INDEX-SEQUENCE corresponding to the given string, viz.

(1) $$\frac{n}{[n]}\ n\ \frac{s}{(n)}.$$

By REDUCING the sub-sequence $\frac{n}{[n]}$ n to n, we obtain the FIRST DERIVATIVE

(2) $$n\ \frac{s}{(n)},$$

from which, by another reduction, we get the SECOND and LAST DERIVATIVE

(3) $$s.$$

Let us notice immediately another important advantage of our notation: we have not only a mechanical method of testing the SYNTACTIC CONNEXITY of a given string but also a mechanical method of finding the so-called constituents of any given syntactically connex string. In the given example, we find by quick inspection that *John sleeps* is not a constituent of *Poor John sleeps* (in spite of the fact that *John sleeps* is connex in itself), since reducing first n $\frac{s}{(n)}$ to s we would arrive at the derivative

(2′) $$\frac{n}{[n]}\ s,$$

from which no further derivation is possible, showing (by the fact that the last derivative, or EXPONENT, in this case does not consist of a single index) that the whole string, analyzed in this way, is not syntactically connex. Strictly speaking, we are only entitled to the following conditional statement: IF *Poor John sleeps* is a syntactically connex string, then *John sleeps* is not one of its constituents.

[4] See *Methods*, ch. 16. Harris uses an equal-sign instead of the arrow, and juxtaposition instead of the arch.

Let us also take notice that a complete element-list of the stated character enables us to synthesize all possible sentences of the given language without any additional rules. Let us consider, for instance, a language that contains elements belonging, respectively, to the categories n, $\frac{\mathrm{n}}{[\mathrm{n}]}$, $\frac{\mathrm{n}}{[\mathrm{s}]}$, and $\frac{\mathrm{s}}{(\mathrm{n})[\mathrm{n}]}$ (the category of strings that form sentences out of a left n and a right n). Then we know that any syntactically connex sequence of elements belonging to n, $\frac{\mathrm{s}}{(\mathrm{n})[\mathrm{n}]}$, $\frac{\mathrm{n}}{[\mathrm{s}]}$, n, $\frac{\mathrm{s}}{(\mathrm{n})[\mathrm{n}]}$, $\frac{\mathrm{n}}{[\mathrm{n}]}$, $\frac{\mathrm{n}}{[\mathrm{n}]}$, n, in this order, would be a sentence, since the only possible exponent of the corresponding index-sequence is s, as can be seen from the following derivation (using, for the sake of typographical simplicity, a slant-line instead of the fraction-bar):

$$\begin{array}{llllllllll}
(4) & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[s]} & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[n]} & \mathrm{n/[n]} & \mathrm{n} \\
(5) & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[s]} & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[n]} & \mathrm{n} & \\
(6) & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[s]} & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n} & & \\
(7) & \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[s]} & & \mathrm{s} & & & \\
(8) & \mathrm{n} & \mathrm{s/(n)[n]} & & \mathrm{n} & & & & \\
(9) & & \mathrm{s.} & & & & & &
\end{array}$$

Whether all these categories have instances in a given language is of course an empirical question. With respect to English, for instance, this question could be answered Yes, roughly speaking. Indeed, *John knew that Paul was a poor man* would be a sentence of the type mentioned. (The qualification 'roughly speaking' is necessary, since it is obvious that the categories dealt with so far are too gross to be applicable to ordinary languages; according to such an application, *Man knew that John was poor a Paul* would also have to be considered a sentence, a thing which most people would hesitate to do.) The same holds for French, where we have sentences like *Jean savait que Paul était un pauvre homme*, but not for German, which lacks elements belonging to the category $\frac{\mathrm{n}}{[\mathrm{s}]}$ (at least on an unsophisticated level, for which *Paul war ein armer Mann* and *Paul ein armer Mann war* are not automatic alternates of the same sentence, but which considers the first string alone as a sentence and the second as syntactically disconnex).

These preliminary considerations should suffice to show that the designation of syntactic categories with the help of such symbols (which carry with them, so to speak, indications of the environments in which members of these categories appear) is a method certainly superior to other prevailing ones in at least one respect, if it can be carried through consistently. Rudiments of this notation already appear in the symbol Vv that was used in our first illustration to designate the category to which *-s* belongs.

Let us begin by stating some assumptions on which our method is based. We assume that with respect to any two given ELEMENT-TOKENS (ink-marks, sounds) it is known whether or not they stand in the relation of EQUIFORMITY, which we take in this context as an undefined primitive relation. Whenever two element-tokens are equiform we shall say that they belong to the same ELEMENT-FORM. The relation of equiformity can be extended, in an obvious way, to hold also between STRING-TOKENS. If all tokens of some form belong to the same category, then this form will be called PURE. If not all tokens of some form belong to the same category, but each token belongs to exactly one of n categories, we shall call the form MIXED and consider it as the (set-theoretical) sum of n TYPES belonging to n different type-categories. A pure form is, of course, its only type. To illustrate: assuming that all tokens of *Paul* belong (in English) to the same token-category (viz. of proper names), then the type *Paul* belongs to the type-category of proper names. Assuming that some tokens of *poor* belong to a certain category, others to another category, still others to a third one, and none to another, we shall consider the form *poor* as mixed and composed of the three types, say $poor_1$, $poor_2$, and $poor_3$.

Observing a given token, we know, in general, to which form it belongs, but not, unless the form is pure, to which type. To find out the type of a token in a given context, we usually have to take account of the linguistic environment of the token and often also of the extra-linguistic context of its production. In all those cases where linguistic environment alone is sufficient to fix the type of a given token, our notation will facilitate this determination and formalize it in such a way that a suitably constructed machine should be able to carry it out. (Even this statement needs qualification: it seems that the proposed notation will be effective only where the crucial environment is not too extended. When the elements are words, the environment taken into account in our notation does not go beyond an utterance.)

According to the envisaged notation, to each element-form there will be assigned a class of n ($\geqslant 1$) symbols denoting the categories of the types to which the tokens of this form belong. These symbols will be called the INDICES of the form and their class the index-class of this form. To a sequence of elements the sequence of their index-classes will be correlated. To arrive at the category-system, we make, among others, the following assumptions. Each sentence that is not an element is regarded as the outcome of the operation of one sub-sequence upon the remainder, which may be to its immediate right or to its immediate left or on both sides. ('Left' and 'right' are to be understood here, as in what follows, only as the two directions of a linear order.) That sub-sequence which is regarded as operating upon the others will be called an OPERATOR, the others its ARGUMENTS. In a two-element sentence, for instance, one element will have to be the operator, the other its argument. In this case, having only one argument, the operator is SINGULARY. In other cases the operator may be BINARY, TERNARY, etc. According to the position of the arguments, we have to distinguish between a singulary right operator, singulary left operator, binary right operator, binary left operator, binary right-left operator, etc. The verbal terminology will soon become pretty involved. We shall therefore use in general the following symbolism: an operator that forms a sentence out of m left arguments belonging (from right to left) to the categories $\alpha_1, \cdots, \alpha_m$, respectively, (the α's may be different but need not be so) and out of n right arguments belonging (from left to right) to the categories $\beta_1, \cdots, \beta_n$, respectively, will be said to belong to the category

$$\frac{s}{(\alpha_m) \cdots (\alpha_1)[\beta_1] \cdots [\beta_n]} \qquad (m + n \geqslant 1)$$

If both the operator and the arguments are elements, then the only remaining thing to be done is to assign the arguments to such categories as will yield an over-all simplest description. (The tremendous problems connected with this procedure cannot be discussed here.) If, however, either the operator or some argument, or both, are PROPER STRINGS, i.e. consist of more than one element, they too have to be regarded as the result of the operation of some sub-sequence upon the remainder. The result of the operation in this case will, in general, no longer be a sentence. Whenever an operator, out of m left arguments belonging to $\alpha_1, \cdots, \alpha_m$ and n right arguments belonging to $\beta_1, \cdots, \beta_n$, respectively, forms a string belonging to the category γ (identical with one of the α's or β's or different from all of them), it will be said to belong to the category

$$\frac{\gamma}{(\alpha_m) \cdots (\alpha_1)[\beta_1] \cdots [\beta_n]} \qquad (m + n \geqslant 1)$$

Elements which are operators in a given string may be arguments in another string or even at another place in the same string, and similarly with arguments. It seems plausible, however, that the requirement of over-all greatest simplicity will necessitate, at least with respect to languages with a finite number of elements, the classing of some types as arguments in all contexts. We shall now make the assumption that the languages we are dealing with contain types which are 'absolute' arguments, so to speak. These types will be said to belong to BASIC CATEGORIES. For purposes of illustration, we shall assume that sentences, proper names, and common nouns belong to basic categories. As the symbol for the category of sentences we shall continue to use s, if necessary with subscripts. As the symbol for the other basic categories we shall use n—again, if necessary, with subscripts. (If the language to which these categories apply is not assumed to be fixed in the given discourse, superscripts can be used to indicate that language relative to which the categorization is to hold.)

Strings which belong to basic categories will themselves be called BASIC. Strings that are not basic are operators, and belong to OPERATOR-CATEGORIES. There is a potentially infinite and elaborately ramified hierarchy of them, but in an effective description of a given language only some of them will be used.

We are now ready to describe that operation upon index-sequences, to be called DERIVATION, on the basis of which the other concepts of our method will be defined. By a derivation of an index-sequence we understand the replacement of any sub-sequence of the given sequence of the form

$$\alpha_m \cdots \alpha_1 \frac{\gamma}{(\alpha_m) \cdots (\alpha_1)[\beta_1] \cdots [\beta_n]} \beta_1 \cdots \beta_n \qquad (m + n \geqslant 1)$$

by γ. The resulting index-sequence is called the DERIVATIVE of the original sequence.

A given index-sequence may obviously have more than one derivative. We already saw that the index-sequence (1) had both (2) and (2′) as its derivatives. Another example is provided by the string *a very poor man*, to which, among others, the following index-sequence is correlated (the use of the double slant-line should be sufficiently clear):

(10) n/[n] n/[n]//[n/[n]] n/[n] n

This has two derivatives:

(11) n/[n] n/[n] n

(11′) n/[n] n/[n]//[n/[n]] n.

A derivative may, in its turn, have one or more derivatives. Thus, (11) has the following as its only derivative:

(12) n/[n] n,

which again has as its only derivative

(13) n,

whereas (11′) has no derivative at all. (11) and (11′) may be called the FIRST DERIVATIVES of (10), (12) its SECOND DERIVATIVE, (13) its THIRD DERIVATIVE, (11′) and (13) its LAST DERIVATIVES or EXPONENTS. (13), but not (11′), is a PROPER EXPONENT, i.e. one consisting of a single index, and any derivation leading up to it a PROPER DERIVATION. The terms 'derivative' and 'exponent' will be used not only with respect to index-sequences but also with respect to the strings to which these sequences are correlated.

Since an element may have more than one index correlated to it, a string may have more than one index-sequence correlated to it. If at least one index sequence of the set of index-sequences correlated to a given string has at least one proper exponent, the string will be called (SYNTACTICALLY) CONNEX. *Poor John sleeps* and *a very poor man* are both connex, each having at least one proper derivation; *poor sleeps John* is not, since neither of its two index-sequences

n/[n] s/(n) n

n s/(n) n

has a proper derivation, as can be verified immediately.

Here, however, arises the following interesting situation. A string that is connex by itself need not be CONNEX AS A SUB-SEQUENCE OF SOME OTHER STRING. Before we define this phrase, let us give an example. *John sleeps* is connex by itself but is not connex within *Poor John sleeps*, so far on an intuitive basis.

Let us now give the strict definitions: A string m_1 will be said to be CONNEX AT A CERTAIN PLACE WITHIN A STRING m_2 WITH RESPECT TO THE DERIVATION d_1 if (1) m_2 is connex, (2) d_1 is proper, (3) d_1 includes a subderivation in which the index-sequence of m_1 at the place in question has a proper exponent. An exact definition of the term 'subderivation' would be somewhat tiresome; it is hoped that the illustration given will make its intended meaning sufficiently clear. In the proper derivation (1)—(2)—(3) above, the index-sub-sequence correlated to *Poor John*, viz. n/[n] n, has the exponent n. *Poor John* is therefore connex within *Poor John sleeps* with respect to this derivation, but *John sleeps* is not connex within *Poor John sleeps* with respect to this derivation.

We now define 'm_1 *is connex within* m_2' as short for 'm_1 is connex within m_2 with respect to all proper derivations of m_2', and 'm_1 *is thoroughly connex*' as short for 'm_1 is connex within all m_i of which it is a (proper or improper) part'.

Clearly, not every connex string has to be also thoroughly connex. In English, *John sleeps* is connex but not thoroughly connex since it is not connex within *Poor John sleeps*. That a language should exhibit this character may be deplored, since it introduces complications into its description and into the analyses carried out on the basis of such a description. We shall take up this point again at a later stage.

The complications mentioned are not such as to cause, by necessity, any major ambiguities. The knowledge that a string is thoroughly connex would indeed dispense with the task of testing whether this string is connex within some given context. That this knowledge is not at our disposal might necessitate more complex checking procedures, but the outcome of these procedures can still be unique. Knowing that *John sleeps*, though connex, is not thoroughly connex, we might be interested in finding out whether it is connex within *Paul thinks that John sleeps*, or at least whether it is connex within this larger string with respect to some of its proper derivations. This last question can indeed be answered in the affirmative by exhibiting the following proper derivation:

	Paul	*thinks*	*that*	*John*	*sleeps*
(14)	n	s/(n)[n]	n/[s]	n	s/(n)
(15)	n	s/(n)[n]	n/[s]	s	
(16)	n	s/(n)[n]	n		
(17)		s.			

The relevant subderivation is framed. But is *John sleeps* also connex with respect to all other proper derivations of *Paul thinks that John sleeps*? The derivation given above is the only proper one with (14) as the original index-sequence. But (14) is only one out of many other possible original sequences. *Thinks* may also have at least the indexes s/(n) and s/(n)[s] (as in *Paul thinks* and *Paul thinks John is sleeping*, waiving possible sophistications) and *that* also has the indexes n and n/[n] (as in *Paul believes that* and *Paul likes that girl*). Disregarding other possible indexes, we have therefore before us at least nine original index-sequences for the given string, which we might arrange in the following way:

Paul	*thinks*	*that*	*John*	*sleeps*
	s/(n)	n		
n	s/(n)[s]	n/[s]	n	s/(n)
	s/(n)[n]	n/[n]		

By systematic testing we can find that only one other original index-sequence out of the possible nine has a proper derivation. The sequence and its derivation are:

(14′)	n	s/(n)[s]	n/[n]	n	s/(n)
(15′)	n	s/(n)[s]	n		s/(n)
(16′)	n	s/(n)[s]		s	
(17′)			s.		

Since *John sleeps* is not connex within *Paul thinks that John sleeps* with respect to this derivation, it is not connex within this larger string (without qualification). In this case, however, we can describe the situation in a slightly more precise way and say that *John sleeps* is connex within *Paul thinks that John sleeps* with respect to a certain index-sequence of this larger string, since *John sleeps* is connex within *Paul thinks that John sleeps* with respect to every proper derivation starting with this original sequence.

The fact that (14′) is a far less likely sequence for *Paul thinks that John sleeps* than (14) is of high importance for what we might call STATISTICAL SYNTAX. Our investigation lies on that level where relative frequencies are not yet taken into account, but only possibilities and impossibilities of occurrence. We also disregard, on this level of approximation, the differences in intonation-patterns which would determine, with high likelihood, whether a given uttered token of *Paul thinks that John sleeps* has the one or the other of the two stated original sequences assigned to it.

Instead of the phrase 'is connex within m_2 with respect to d_1' we shall, in general, use the more customary expression 'is a constituent of m_2 with respect to d_1', hence instead of 'is connex within m_2' also 'is a constituent of m_2'. We can now also define the phrase 'is an immediate constituent of m_2 with respect to d_1' as meaning 'is a constituent of m_2 with respect to d_1 but is not a constituent of any m_3 that is a (proper) constituent of m_2 with respect to d_1'. In our last example, for instance, *that John sleeps* is an immediate constituent of *Paul thinks that John sleeps* with respect to the derivation (14)—(17), and also with respect to the derivation (14′)—(17′), and this in spite of the fact that it has different exponents in these two derivations.

With respect to the first derivation, the immediate constituents of *Paul thinks that John sleeps* are *Paul*, *thinks*, and *that John sleeps*. Of these, the first two are elements and the third a proper string which has, therefore, immediate constituents of its own, with respect to the same derivation, viz. *that* and *John sleeps*, of which the first is an element and the second again a proper string with the immediate constituents *John* and *sleeps*. In a self-explanatory terminology, we may therefore say that, with respect to the given derivation (14)—(17), the elements *Paul* and *thinks* are IMMEDIATE CONSTITUENTS of the given string, *that* is a CONSTITUENT OF THE SECOND ORDER, *John* and *sleeps* are CONSTITUENTS OF THE THIRD ORDER; and we may say that the string itself is OF THE THIRD ORDER. A connex two-element string would then be of the first order with respect to any of its proper derivations, and we might say, if this should prove to be convenient, that any element is OF ZERO ORDER (without qualification).

If two tokens of the same element-form occur in a given string, then explicit reference to these occurrences may have to be made, since even if they belong to the same type they need not be constituents of the same order with respect to a given proper derivation. It is clear that m_1 may be an immediate constituent of m_2 with respect to some proper d_i but not with respect to some different d_j. If, however, m_1 happens to be an immediate constituent of m_2 with respect to all proper derivations, then we shall drop the qualifications and say that m_1 is an immediate constituent of m_2. Under the assumption that the set of index-sequences given above for *Paul thinks that John sleeps* is exhaustive, *Paul*, *thinks*, and *that John sleeps* are immediate constituents of this string, without qualifications.

Many of the concepts defined so far are much more dependent than one might at first thought assume upon the specific derivation, and hence upon the form of the original index-sequence. One might tend to believe, for instance, that it would make no appreciable difference whether one regarded a given operator as forming a sentence out of a right n and a left n, i.e. as an operator s/(n)[n], or as forming out of a right n an operator that forms a sentence out of a left n, i.e. an operator s/(n)//[n]. But such a belief would be a mistake. It makes a considerable difference in the organization of immediate and other constituents whether we treat *loves* (say) as an operator which out of a left n *John* and a right n *Mary* forms a sentence, *John loves Mary*, IN ONE COMPLEX STEP, or as an operator which out of a right n *Mary* forms an operator, *loves Mary*, which out of a left n *John* forms a sentence IN TWO SIMPLE STEPS. According to the second treatment, *loves Mary* is an immediate constituent of the whole; according to the first, it is no constituent at all. The fact that being-a-constituent-of is a relation which is not invariant even with respect to such 'inessential' transformations as that of s/(n)[n] into s/(n)//[n] shows that this relation and its cognates are of somewhat restricted importance. Incidentally, according to the Aristotelian analysis, *John loves Mary* has to be understood as a subject-predicate sentence with *John* and *loves Mary* as its immediate constituents. The categorization which treats *John* and *Mary* on a par, as respectively the left and right arguments of *loves*, though much in favor with modern logicians, for a long time was not regarded as proper.

But leaving philosophical and logical considerations aside, it is an interesting problem to compare the advantages and disadvantages of using only SINGULARY OPERATORS WITH COMPLEX NUMERATORS as against using only n-ARY OPERATORS WITH SIMPLE DENOMINATORS, or using both simultaneously. In certain presentations of COMBINATORIAL LOGIC,[5] for instance, singulary operators are preferred, in spite of the fact that this involves multiplying the number of operations. Nothing more will be said here on this topic.

It is useful, in certain investigations, to distinguish between operators which

[5] Cf. Haskell B. Curry. A theory of formal deducibility, *Notre Dame mathematical lectures* No. 6, 1950.

out of their arguments form a string belonging to the same category as the arguments, and those which do not. The first kind might be called ENDOTYPIC, the second EXOTYPIC. That type of *poor*, for instance, which belongs to the category n/[n], is endotypic, while *sleeps*, which belongs to s/(n), is exotypic. In certain contexts, it might be profitable to use a slightly different classification and to regard operators belonging to categories of the form $\alpha/(\alpha)$ ... $(\alpha)[\alpha]$ ... $[\alpha]$ as endotypic. That type of *and*, for instance, which is an s/(s)[s], would be endotypic according to the second conception but exotypic according to the first. We might perhaps, if necessary, distinguish between endotypic in the narrower sense and endotypic in the wider sense.

English adjectives when used in adjectival function, demonstratives when used in adjectival function, articles, adverbs, and conjunctions are endotypic; verbs and prepositions are exotypic; nouns in general are neither, being mostly basic. This is only a rough application of our terminology, since it is obvious that it is unable, as developed so far, to cope with the whole gamut of relationships that exist between the elements in English or in any other natural language. With this provision in mind, adjectives will in general belong to n/n-categories (omitting parentheses and brackets now for the sake of simplicity), as will articles and adjectival demonstratives; conjunctions will be s/ss (hence endotypic only in the wider sense); adverbs will be n/n//n/n (VERY *good*), s/n//s/n (*sleeps* SOUNDLY), s/nn//s/nn (ARDENTLY *hates*), s/s (UNFORTUNATELY *John died*), etc.

To get a slightly better outlook on the effectiveness of the proposed notation, let us analyze a string with a much more complex structure than that of the strings so far, though still very far from the top of the complexity ladder. The string is taken from a language[6] which most readers will not know. Its rough transliteration is:

1	2	3	4	5	6	7	8	9
moše	*yada*	*ki*	*pinxas*	*xaxam*	*yoter*	*meašer*	*axoto*	*haktana*

Let us assume that by looking up the elements of these strings in the category-list (so far non-existent), we obtain the information given in Table 1. (The list in Table 1 is incomplete even with respect to that very rough approximation which we have set as our standard.) We have before us a set of 216 original index-sequences. Systematic testing for SUITABLE original sequences (i.e. sequences with proper derivations) would be laborious though perfectly feasible for a properly designed machine. We shall use some shortcuts. It is easy to realize that 9a does not fit. 5a does not fit either. Similarly 5b, 6c, hence also 7c, are out. The exponent of 7–9 obviously operates on the exponent of 4–6 and must be s, because of 3. 3b is unsuitable, as is 2a. Exactly one suitable original index-sequence is left:

$$(20)\qquad \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{n} & \mathrm{s/(n)[n]} & \mathrm{n/[s]} & \mathrm{n} & \mathrm{s/(n)} & \dfrac{\mathrm{s/(n)}}{\mathrm{(s/(n))}} & \dfrac{\mathrm{s/(n)}}{\mathrm{(s/(n))[n]}} & \mathrm{n} & \mathrm{n/(n)} \end{array}$$

[6] Colloquial Hebrew. For the sake of the simplicity of analysis, however, the common *meaxoto* has been replaced by the (colloquially) much less frequent *meašer axoto*.

Many derivations start from (20). There are already three different first derivatives as a result of operating 5 upon 4, 6 upon 5, and 9 upon 8. It is, however, easy to see that the first derivation leads into a blind alley. We thus arrive at the interesting (though on second thought not surprising) result that only two proper derivations are correlated to the given string, even though it has 216 original index-sequences, many of them with more than one first derivative. We present one of the two proper derivations in the abbreviated scheme of Table 2. The second derivation differs from the one presented in Table 2 only in that steps

	1	2	3	4	5	6	7	8	9
a	n	s/(n)	n/[s]	n	n	$\frac{\text{n/(n)}}{\text{(n/(n))}}$	$\frac{\text{n/(n)}}{\text{(n/(n))[n]}}$	n	n
b		s/(n)[n]	s/[s]		n/(n)	$\frac{\text{s/(n)}}{\text{(s/(n))}}$	$\frac{\text{s/(n)}}{\text{(s/(n))[n]}}$		n/(n)
c					s/(n)	$\frac{\text{s/(n)[n]}}{\text{(s/(n)[n])}}$	$\frac{\text{s/(n)[n]}}{\text{(s/(n)[n])[n]}}$		

TABLE 1

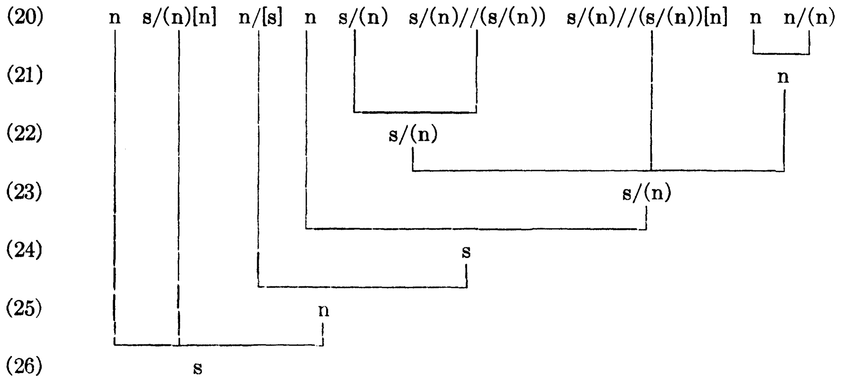
(20)
n s/(n)[n] n/[s] n s/(n) s/(n)//(s/(n)) s/(n)//(s/(n))[n] n n/(n)
(21)
n
(22)
s/(n)
(23)
s/(n)
(24)
s
(25)
n
(26)
s

TABLE 2

(21) and (22) change places, a difference which we may safely describe as trivial. If the category-list were constructed so as to contain singulary indexes only, the derivation would have contained two more steps, each THREE-FORK being split into a sequence of two TWO-FORKS, and the set-up of immediate and *n*-order constituents would have undergone some changes.

I am fully aware of the inadequacies of the proposed notation. Its shortcomings are many and of various kinds. Some are due to the effort of presenting the main ideas as simply as possible, and can easily be overcome through a less simplified approach. Moreover, the problem of categorization is far from a satisfactory solution, and the proposed notation as such (I repeat) cannot put the linguist in a better position for solving it, though it may well redirect his attitudes.

Two further points should be mentioned. First, we have said nothing about what the linguistic elements are to be in specific cases; in our examples, we used words. It is plausible that other elements might be more suitable, if not for all languages, at least for many. Phenomena like separable prefixes and Harris's 'long components' will pose additional problems.

Another feature, related to the previous, is our deliberate restriction of the range of arguments to the IMMEDIATE environment of the operators. This will prove to be disturbing with regard to languages that have constructions like the English string *Paul strangely enough refused to talk,* where *strangely enough* is probably best regarded as an s/s operator, since it is the whole event which is regarded as strange and not the way in which Paul refused to talk. But whether we assign s/(s) or s/[s] to this string, it will turn out that the larger string will be disconnex, as the reader can verify for himself. However, both *Paul refused to talk* (,) *strangely enough* and *Strangely enough* (,) *Paul refused to talk* will come out all right. The last two sentences are commonly regarded as variants of the first, with hardly any difference in meaning—certainly not in cognitive meaning.

Though this example points to a certain shortcoming of our notation, it also shows that only a small change is needed to make it adequate for handling complications of this type. Using certain auxiliary symbols such as commas would change the picture. If we write *Paul, strangely enough, refused to talk* (which is, incidentally, the common usage), and interpret the function of the commas as giving us license to lift the string between them from its position and deposit it at some other position (within certain limits, of course), we can still adhere to the simple rules of immediate environment. It remains to be seen whether devices of such a simple nature will enable us to retain a notation which takes account only of the immediate environment with respect to all languages.