Research Group for
Quantitative Linguistics
Fack
Stockholm 40
SWEDEN

KVAL PM 337
June 19, 1967

SLANT GRAMMAR CALCULUS

By

HANS KARLGREN

The work reported in this paper has been sponsored by Humanistiska forskningsrådet, Tekniska forskningsrådet and Riksbankens Jubileumsfond, Stockholm, Sweden.

SLANT GRAMMAR CALCULUS

By

HANS KARLGREN

KVAL, Fack, Stockholm 40, Sweden

Summary

Bar-Hillel and Lambek have outlined a syntactical description where the syntactical category symbols are written as fractions and where the analysis of a given sentence is performed according to rules very similar to common arithmetical reduction of fraction expressions with factors that cancel out. Are there algorithms for applying such a calculus to normally complex natural language structures?

1. Aim

We seek a formal recognition procedure that will enable us to decide for any given sequence of elements from a given language whether or not the sequence is grammatical.

We consider only one method, named that of a categorial grammar (Bar-Hillel and Lambek).

We make the following assumptions:

(a) The knowledge we want to utilize for recognition can without residue be summarized in

> a list, giving for each word the grammatical categories the word belongs to; a set of combination rules for the category symbols.

(b) A sequence of elements is grammatical if there exists at least one word-for-word translation of it into grammatical category symbols which yields a symbol sequence that is permitted according to the set of combination rules. We say that a symbol sequence which agrees with the combination rules is a grammatical symbol sequence.

2. Shrinking Procedure

We assume that it is possible to verify the grammaticality of a symbol sequence by reduction of it to simpler and shorter sequences step by step. In each step one or more symbols in the sequences are replaced by one new symbol.

The string replaced by one other symbol will - to begin with without linguistic interpretation - be called a syntagm; the replacing symbol will be called the name of the syntagm.

The work reported in this paper has been sponsored by Humanistiska forskningsrådet, Tekniska forskningsrådet and Riksbankens Jubileumsfond, Stockholm, Sweden.

By successive application of rewriting rules, the original sequence is shrunk to a no longer reducible residue, which may be just one symbol. If this residue is contained in a given list of permissible sentence patterns, the sentence is grammatical.

3. Slant Grammar Calculus

The kind of grammar under study we shall simply call slant grammar from its salient trait, the notation. It is characterized by the following properties:

a) The category symbols are all of the form

a (atomic symbols)

or $\left.\begin{matrix} x/y \\ y\backslash x \end{matrix}\right\}$ (complex symbols)

where x and y in their turn have the same form (atomic or complex) as the category symbols. We shall call a and x numerators and y a denominator in such cases.

b) Combinatorics is condensed to the following

α) a symbol sequence is a grammatical syntagm of type t if and only if it can be reduced to t by successive application of one of the following two cancellation rules for contracting two neighbouring symbols of the - original or the so far reduced - sequence into one symbol:

$$x/y\ y \rightarrow x$$
$$y\ y\backslash x \rightarrow x$$

where x and y are atomic or complex symbols.

β) a grammatical sentence is a syntagm of a type which belongs to a short list of possible types of patterns, say type s.

The categorial notation seems helpful in establishing a recognition calculus. Some programmable algorithms will be discussed in this paper.

It is easily seen that slant grammars of the type discussed are equivalent to context-free phrase structure grammars (as far, i.e., as any generative grammar can be "equivalent" to a recognition grammar).

The cancellation rules presuppose that if the symbols a/b and b are reduced to a, there must not stand anything between the syntagms a/b and b - i.e., there must be no hole in the syntagm a - although the symbols a/b and b may in the original sequence stand widely apart.

A slant grammar for one given language may be written in many different ways. Thus one may design the grammar so that the category symbols have at the most one denominator and even so that they have only left denominator or only right denominator (Marcus).

A natural way to design the grammar would be to let governed syntagms have simple symbols (y) and the governing ones complex symbols x/y, or inversely, so that the relation operator/operand would imply dependency relation.

However, the number of alternative symbols for each word will tend to increase if such a priori rules should apply to the whole set of symbols. Given the algo-

rithm for recognition, one may ask how the categorial grammar should be designed so as to give the minimum number of operations, e.g., so as to yield on an average, the minimum number of possible word-for-word translations into grammatical symbols.

4. Reduction Procedure

To begin with, we shall investigate some procedures for analysis of a given sequence of category symbols. Then, cf. 8 below, we turn to the practically more important problem when not a sequence of symbols but a sequence of words is given, each word having several potential categories.

5. Substituting Complex Symbols

We make a preliminary simplification of the problem by replacing every complex denominator in the sequence by a new, arbitrary atomic symbol. Simultaneously, we make corresponding substitutions of numerators: if we replace b/c by x as a denominator, we also replace b/c by x at some other place, where b/c appears as a numerator.

Example:

b/(c/a) c/a b\a/(b/c) d d\(b/c) ⇒ b/y y a/x d d\x.

Now, if b/c should happen to appear in numerator position more often in the given sequence than it does qua denominator, this replacement can be performed in more than one way. We then do perform it in more than one way, thus generating a number of alternative symbol strings to be processed. Through this artifice, we have sequences where all denominators are certain to be atoms, a fact which radically simplifies the analysis. Instead we have made the symbol selection procedure more difficult.

Since now all denominators are atoms and since (b\a)/c is equivalent to b/(a/c) and to b\a/c, the brackets are now redundant and can be omitted.

The symbols, then consist of a kernel atom, possibly neighboured at one side or both by a slant and another atom, in its turn possibly neighboured by slant plus atom, and so on, all slants to the left of the kernel being tilted to the left and those to the right tilted in the opposite direction:

a, a/c, b\a/c, g\f\e\d\a/b/c/, ...

If one knows which element is the kernel, one does not even need the slants and we can proceed to simplify one step further:

a, ac, bac, gfedabc, ...

where the underlined characters are numerator atoms and all others are denominator atoms.

(If the language has no rules for the relative order of syntagms, grammaticality is rapidly tested. Just check that to each denominator atom corresponds one numerator atom of the same name, leaving without a match just one numerator, which then denotes the type of the syntagm. In this case we are permitted to treat the atoms as numerators and denominators in the arithmetical sense. If we assign prime numbers to each atom and reduce in the standard arithmetical way, we end up with the numerical value of the type of the syntagm. This simple test may be worth considering as a first check, even though the structure of the language be far more complex.)

6. Wanted: Non-Intersecting Vaults

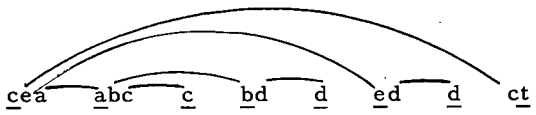
The two above-mentioned cancellation rules are, rewritten in the modified (simpler) notation, fused into one:

> If in a string of characters, an underlined character and a non-underlined character appear, separated only by a space character and denoting the same atom, erase all three characters.

Our task may also be formulated thus:

> Draw a connecting line from each denominator in the string to a numerator with the same name, in the correct direction - i.e., not over the numerator of the own symbol - like vaults over the strings, in such a way that no two such vaults intersect and that one numerator - say t - remains untouched by any vault nor roofed by one. If this succeeds, the string is a grammatical sequence of type t.

Example:



7. Identification of the Correct Numerator

If all numerator symbols are different the task is trivial. If several numerator atoms carry the same name, we have to decide which one is intended by a given denominator. How do we do this?

The problem is in this form a purely computational one - and no easy such problem.

8. Algorithm with Stacking

One algorithm can be summarized as follows:

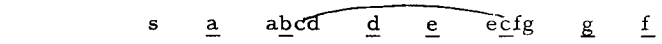
We have a given string of words and want to ascertain whether its type is one of the set T = {s, t, ...}. For each word we have a given set of alternative category symbols.

We first consider the simpler problem of analyzing a given string of category symbols and see if it can be reduced to s.

We join the (not underlined symbol) s followed by space to the beginning of the given symbol string. When the reduction rules are applied the resultant string should vanish; we say it is reduced to unity.

The string now contains exactly as many, say n, numerator and denominator atoms. Every numerator should be paired with one other denominator; the whole problem is to decide which denominator.

Whenever we thus pair two atoms, the intervening string of atoms should vanish under reduction, be reducible to unity. The same holds for the rest of the original string, after the two paired atoms and everything between them has been removed:



In general, then, when one wants to test whether a pairing is a good match or not, one is left with two simpler, isomorphic problems, those of reducing two strings to unity:

s a abfg g f; d d e e

Normally we cannot decide beforehand which pairing to test; we must try alternatives. For each arbitrary choice of procedure, we store the alternatives in an OR-stack, For each pairing we place the bracketed-out string in an AND-stack, to be handled later when and if the string on our working table vanishes. If we have emptied the table and the AND-stack, the given string was reducible to s. If we have an irreducible residue on the table, we clear the table and try the next alternative if there is one in the OR-stack, deleting what has been stored in the AND-stack since the last arbitrary choice; if we cannot reduce to unity the string on the table and the OR-stack is empty, we must give the task up.

To minimize the number of pairings to test for each string we analyze, we draw up a binary matrix M of potential matches, with $m_{ij} = 1$, if the denominator No. i and the numerator No. j are the same character and are placed in proper order and reasonably wide apart in the string. (E.g., if i is a "non-recursive" type of syntagm, "reasonably" placed means that j is the nearest numerator written with the same character as i and placed on the proper side of i.)

If any row or column in M is empty, the task is hopeless. Otherwise, we select the atom whose row or column contains the least number of 1's.

To analyze a string of words we assign to each word one symbol of the type ABCDE, where C is the set of numerators in all the word's category symbols, B is the set of nominator atoms appearing immediately before the numerator in any one of the word's category symbols, etc. On the string of these new symbols the above procedure, mutatis mutandis, is applied. Thus, instead of the condition in (1) above that two atoms should be the "same" characters, it is required that one underlined atom A and a non-underlined atom B appearing separated only by space should fulfill the condition $A \cap B \neq 0$.

References

1. Y. Bar-Hillel, A quasi-arithmetical notation for syntactic description. Language 29, 47-58 (1953).
2. J. Lambek, On the calculus of syntactic types, in "Structure of Language and Its Mathematical Aspects". Proc. 12th Symp. Appl. Math., American Mathematical Society, Providence, R.I., 1961, pp. 166-178.
3. S. Marcus, Algebraic Linguistics; Analytical Models, New York & London, 1967.