MULTI-INDEX SYNTACTICAL CALCULUS

Hans Karlgren*

Introduction

In our work on analyzing Swedish nominal phrases as they appear as document titles - particularly titles of articles in periodicals - we have primarily utilized context-free rules. In an endeavour to reduce the cumbersomeness of such rules, we have used the notation:

(1) $a_{xy}\ b_{xy} \rightarrow c_{xy}$ for x = p, q, r and y = u, v

as a shorthand for six substantially similar rules. The gain is not merely that of avoiding scrivener's palsy - and puncher's impatience, since the analysis program also accepts this shorthand - but also that of clarifying the parallelism between the rules. The rule schema reads "a syntagm of type a combines with one of type b to form one of type c, each being respectively of subclass p, q or r and u or v". If the subscripts are interpretable as linguistic categories, this notation seems quite natural. We might write a fundamental rule of Latin grammar, by way of illustration, thus

$$adj_{ngc}\ \ nom_{ngc} \rightarrow nom_{ngc}$$

which would mean that to a nominal group may be joined an adjective of the respective number gender, and case without changing the syntactical category of the group.

---

* KVAL, Fack, Stockholm 40.
The work reported in this paper has been sponsored by
The Bank of Sweden Tercentenary Fund and
The Swedish Humanistic Research Council

This notational little device actually often reduces the intuitive need for context-sensitive rules, since it performs what these rules are required to do in the domain where we have a choice, namely to bring out the common pattern and leave aside for later consideration the minor adjustments.

Now, in practice, we have for each word or syntagm not one subscript but a set of alternative subscripts. On the initiative of Gunnar Ehrling,* who wrote the analyzer, we further reduce the notation by giving a name to all such sets of alternatives and by specifying in a "multiplication table" the name of the set of alternatives forming the intersection between any pair of such sets. Thus, in place of (1) our rules actually read

(2) $$a_{ik}\ b_{jl} \rightarrow c_{i\cap j\ ,\ k\cap l}$$

where the values of $i\cap j$ and $k\cap l$ are taken from the "multiplication table".

We now ask what will happen if we generalize this index "multiplication" so that it will represent not intersection of index sets but an arbitrary binary operation on the set of index symbols. Particularly, we are interested in the case where this multiplication is non-associative and the set of index symbols is not closed under multiplication. This would mean that the restrictions imposed by the indexes on the sentence or part thereof could, in their turn, be written as a context-free - not a finite-state - grammar over the index symbols.

When the subscript multiplication rules are generalized so far, they are of the same kind as the "multiplication" on the main level, and we prefer to write $a'i'k$ for a_{ik} and we define

* KVAL, Interim Report No 13,
Program för grammatisk analys av texter

multiplication of such index vectors as "inner" multiplication, that is, the corresponding elements are multiplied:

$$a'i'k \quad b'j'l \rightarrow ab'ij'kl$$

We note that, in general, these rules cannot be reduced to a finite list of common context-free rules, as could rules like (1) and (2). For if we can replace ab by c, we may well be unable to replace ij by anything shorter than ij, the multiplication table being blank for ij or even having no row i or column j, since i and j may, in turn, be strings and not elements in the index set. And if the well-formed sequences of indexes are defined by a general context-free grammar and not by a finite-state one, we cannot remedy this by adding more symbols to the index set: the set of triples i, j, ij may then be infinite.

This paper is an attempt to investigate this problem, elaborating such a multi-index calculus a little. First, however, we may be excused for making a summary of the background of the recognition grammar problems for which such a calculus may be useful. The reader who expects to be bored by such a survey should turn directly to page 10 below.

Reduction systems

We introduce some definitions. The terms employed largely coincide with those of current generative linguistics, but some minor adaptions have been made to make the terms adequate for describing the kind of recognition grammars with which we are concerned.

We consider strings over an alphabet $S = \{ a, b, c, . . \}$. We write ab for the string formed by concatenation of two letters a and b, and $\alpha\beta$ for the concatenation of two strings α and β. Concatenation is considered a reflexive, associative but not commutative relation.

We write M for the set of all concatenations of strings in a set M:

$$M^{*} = M \cup \{ \mu\mu^{*} \mid \mu \in M, \ \mu^{*} \in M^{*} \}$$

A rewriting rule, $\alpha \rightarrow \beta$ is a rule which permits us to replace the string α in any string where it may occur by the string β. A reduction rule is a rewriting rule which does not increase the number of words in the string. A reduction system is a set of reduction rules:

$$R = \{ \alpha \rightarrow \beta \mid \alpha \in a_1 a_2 \ldots a_n, \ \beta \in b_1 b_2 \ldots b_n, \ a_i \in S, \ b_j \in S, m \leq n \}$$

By means of R we can define a derivability relation over S^{*}. We say that α is reducible to β, $\alpha \rightarrow \beta$, according to R, if there is a succession of applications of rules in R by which α can be rewritten as β. We include the case where no rule is applied so $\alpha \mapsto \alpha$ for all α. Thus, "$\mapsto$" is a reflexive and transitive relation.

We now define a reduction grammar $G = < S, R, I, T >$ as a specification of a set of strings, a language, over an input alphabet $I \subset S$:

$$L = L(< S, R, I, T >) = \{ \sigma \mid \sigma \in I^{*}, \ \sigma \mapsto \tau \in T \subset S^{*} \}$$

where T is a set of - terminal or, to avoid diametrically opposite associations - target symbols. We say τ is an R-reduction of σ.

Finite Rewriting Systems

Constituent structure grammars and grammar components

We first consider grammars where S is a finite set. We call these grammars constituent structure grammars.

If T contains one single element, say s for sentence, the grammar is a decision grammar, which specifies for each input string whether or not it is grammatical.

Trivially, T can be extended to include a few elements, say s for statement, q for question, and so on. Naturally, we can reformulate a grammar with $T = \{t_1, \ldots, t_n\}$, where n is finite, into a grammar with a unique target element, merely by adding one element, say s, to S and incorporating a few rules $\{t_i \rightarrow s \mid i = 1, \ldots, n\}$ to R.

However, allowing T to be an infinite set is not necessarily a trivial extension.

Trivial but occasionally practical is to define a language $L(S, R, I, A^*)$ where the targets are all the strings over an output alphabet $A \subset S$.

If T is some non-trivially defined subset set, L' of strings over a subset A of S, we have

$$L = L(S, R, I, L')$$

where L' must be defined by some grammar $G' = \langle S, R, A, T \rangle$ We say that $G'' = \langle S, R, I, A \rangle$ is a grammar component and note that G'' and G' together completely specify L. We shall come back to this concept later when we describe more complex grammars as combinations of simple ones.

With the restriction imposed on the rules of R that the right hand side should never be longer than the left hand side, it is obviously always possible in a finite number of steps to decide whether or not a given finite string is reducible to some

element in T, i.e., whether or not it is an element in the set L. For if the given string σ contains m symbols and S contains n different symbols, σ can be shortened at most (m - 1) times and after the i:th time it has been shortened, (i = 0, 1, ..., m - 1), it can be rewritten without shortening at most $(n^{m-i} - 1)$ times without being rewritten as σ, which can always be avoided by keeping a finite record of historical information.

Disjoint constituent grammars

1. A reduction rule where the right hand side contains exactly one symbol is called a context-free rule. If all the rules are context-free we say the grammar and the language is context-free.

If the grammar is context-free we may give it the following interpretation. Let the letters of I be sets, "categories", of strings of linguistic signs. Let ab mean the set of strings consisting of one string contained in category a followed by one contained in b. Let the reduction rules mean inclusion so that, e.g., $ab \subset c$ means that the set ab is included in the set c.

A string σ over I then represents a grammatical sentence of type t, if and only if, $R \Rightarrow \sigma \subset t \in T$.

2. A context-free constituent grammar, then, can be adequately described as a classificational system with finer and broader terms where all classes can be written as concatenations - interpreted as the set of concatenations of the cartesian products - of a finite set S of categories. The process of analyzing sentences of such a language can be performed as a classificational procedure and the result is adequately and exhaustively

statable as the class adherence of sets of successive substrings, representable, e.g., by a tree with no crossing branches.

One may note that the character of a context-free language well conforms with what used to be defined as agglutinative languages, that is with the agglutinative languages as they were commonly defined, not as any existing natural language of any particular group.

The assumptions behind an attempt to describe a real language by a context-free grammar, therefore, are very strong. It is not astonishing that these attempts partially fail; it is astonishing that they have carried as far as they have. For instance, there is no convincing empirical evidence that a decision grammar for a natural language cannot be written as a context-free grammar, though there are ample theoretical reasons not to stake too much on the prediction that no practical counter-examples will turn up in the future.

3. If we add to our context-free grammar rules of the type

$$ab \rightarrow bc$$

or, generally, permutation rules where the same elements recur on the right, though in different order, we broaden, of course, the family of languages under considerations and the interpretation above under 2. no more holds true. But all what was said about the highly specialized character of the languages remains true, except that class adherence is now not confined to sets of successive substrings; the language is characterized by the existence of discontinuous constituents, and except that the tree drawn will have crossing branches here and there. But it is still possible to assign each substring to exactly one immediately

higher order constituent and it is still possible to draw a tree.

We may summarize the constituent so far mentioned under the name disjoint-constituent grammars, i.e., grammars where each constituent is either disjoint from or included in another and where, accordingly, the constituents can be defined as a hierarchial set of equivalence classes over the substrings of the given input string.

Such a classification of substrings is called a p-marker. The hope of expressing the essence of the syntactical structure of a sentence by one p-marker therefore implies strong assumptions about the language.

Overlapping constituent grammar

If the rules of R do not obey the restrictions mentioned for disjoint-constituent structure grammars, that is, if rules occur of the type

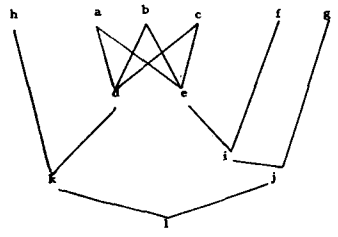
abc → de

or

abc → dc

no equivalence classification of substrings is obvious and no tree can be drawn without further assumptions.

The most natural would be to draw a graph of the following kind:



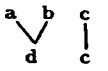
Unlike p-markers, this graph attributes one and the same substring of the input string to more than one higher constituent also when these higher constituents are disjoint. Here abc belongs to d and to e, to k and to i.

It is by no means an unnatural description of a sentence to let one segment have more than one function, nor is it impractical to represent such structures as graphs. On the contrary, that is what graphs are for, and in the special case where no two branches ever coalesce, the graph seems to be so utterly simple that it is, at any rate, rather a waste of paper to print drawings of it.

For a subset of the grammars now under discussion we can, with some good will, construct p-markers, although the same rules contain more than a single right handed element. If the rules are of the type

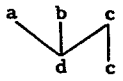
$$abc \rightarrow dc$$

or, generally, only one symbol on the right is different from the corresponding symbol to the left, we may, by convention*, consider ab to be a constituent of type d, whereas c only functions as a context. For these context-sensitive cases we therefore can agree to represent our reduction as follows:



instead of

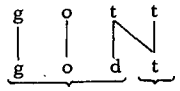
It might seem as natural to draw



saying that <u>d</u> is a representation of <u>c</u> as well as of <u>ab</u>, since d could not have been rendered as <u>ab</u> unless <u>c</u> had been present.

* Chomsky (1963) p. 294, Handbook of Mathematical Psychology, edited by Luce, Bush, and Galanter.

One would then have overlapping constituents in cases such as Swedish gott, reducible to godt:



Nobody seems to be over-happy with this attempt to "add conditions to guarantee that a p-marker for a terminal string can be recovered uniquely from its derivation" and for this and more serious reasons linguists turn away from these types of constituent grammars altogether. But it is characteristic that one attempts to find "unique" equivalence classifications, i.e., tree graphs of the simple kind described. "We assume that such a tree graph must be a part of the structural description of any sentence; we refer to it as a phrase-marker p-marker. A grammar must for adequacy provide a p-marker for each sentence".* In other words, rather than modify the kind of graph employed, one replaces it, in transformational grammar, by an ordered set of such simple graphs.

The multi-index notation permits an alternative mode of presentation, as will appear in the next few paragraphs.

Infinite Rewriting Systems

We now consider the case where a grammar $G = < S, R, I, T >$ contains an infinite alphabet S.

In particular, we consider the set S of vectors over a finite set S' of indexes:

$$S = S' \cup \{ s_1 ' s_2 ' \ldots ' s_n \mid s_i \in S' \}$$

* Chomsky, op. cit. p. 288.

For S we introduce the general multi-index multiplication schema:

(1) $$(s_1 ' s_2 ' \dots ' s_n)\,(t_1 ' t_2 ' \dots ' t_m) \rightarrow$$

$$(s_1 t_1)' (s_2 t_2)' \dots ' (s_n t_n)' t_{n+1} ' \dots ' t_m \quad \text{if } n < m$$

$$(s_1 t_1)' (s_2 t_2)' \dots ' (s_n t_n) \quad \text{if } n = m$$

$$(s_1 t_1)' (s_2 t_2)' \dots ' (s_m t_m)' s_{m+1} ' \dots ' s_n \quad \text{if } n > m$$

that is, for $i > n$ and $j > m$ we consider $s_i = t_j = e$, where e is a unit element such that $ae = ea = e$ for all a.

R' contains, except the general multi-index schema (1), a finite set R' of rules or rule schemata over S

(2) $$R' = \{\alpha \rightarrow \beta \mid \alpha = a_1 a_2 \dots a_n,\ \beta = b_1 b_2 \dots b_m,\ n \leq m\}$$

where a_i and b_j are elements in S or variables over S or over specified subsets thereof.

T is given either explicitly or as an infinite subset of S

$$T = \{t ' x \mid t \in A \subset S,\ x \in S\}$$

i.e., as those elements in S which consist of an element in a finite set A, arbitrarily subscripted.

We note that every element s in S defines an infinite class of elements beginning with the vector s, just as a decimal number defines a class of number with the same or a greater number of digits.

The rules of R are such as

1 $ab \rightarrow c$

2 $a ' x\ \ b ' y \rightarrow c ' z$

3 $a ' x \rightarrow b$

4 $a \rightarrow b ' x$

and so on. To make a language decidable it is obviously sufficient - by way of analogy with the reasoning above - to require that the right-hand side should never contain more letters out of the alphabet S' than the left-hand side, thus excluding rules like rule 4 above. The fact that the letters are here distributed over different levels, so constituting one or more symbols of S, cannot invalidate that argument.

The conclusion obviously also remains intact if we accept rules with a longer right-hand side for rewriting symbols which never occur on the right-hand side of any rule, that is, if we make allowance for assignment rules.

In the following we shall restrict ourselves to <u>context-free</u> multi-index rules, that is, the rules shall

a) contain one element of S on the right-hand side and wherever practical* the rules shall also
b) contain at most as many elements of S' on the right-hand side as on the left-hand side, except where the left-hand side consists exclusively of elements which occur on the right-hand side of no rule.

Though each rule is a context-free rule, such a multi-index grammar is not a disjoint-constituent grammar; constituents do overlap:

Let us consider a grammar where

$$ab \rightarrow d$$

$$dc \rightarrow s$$

$$xy \rightarrow u$$

$$uz \rightarrow v$$

and where $s'v \in T$. Let us consider the analysis of the string $a'x\ b'y\ c'z$:

* The second restriction is unnecessarily severe. One may well include, e.g., r ules which are not reductive with reference to S'' but which are <u>strictly</u> reductive on the highest level they refer to and which do not increase the number of levels referred to by any rule.

$a'x \quad b'y \quad c'z$

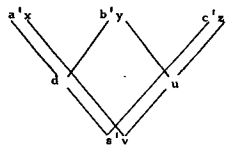
$d'xy \quad c'z$

$s'xyz$

$s'xu$

$s'v$

or graphically:



We see that segmentation is overlapping but that each level of indexes represents one equivalence classification and one tree-shape graph.

In many cases, context-free multi-index rules are weakly equivalent to context-sensitive rules, as will appear from the following few examples of languages which notoriously cannot be described with ordinary context-free rules. Crudely, we may say that taking an index on another level into account is an implicit way of regarding context.

Example 1. The language "$a^n b^n c^n$".

R' : $a \rightarrow x'p$

$b \rightarrow y'p$

$c \rightarrow z'q$

$xy \rightarrow s$

$xsy \rightarrow s$

$sz \rightarrow s$

$ppq \rightarrow e$ where e is the unity element.

Illustration :

aabbcc

x'p x'p y'p y'p z'q z'q

x'p s'pp y'p z'q z'q

s'pppp z'q z'q

s'pp z'q

s'e $\underline{=}$ s

T = s

Example 2. The "reduplication" language, consisting of an arbitrary string of a's and b's followed by the same string repeated.

R : $xy \rightarrow x'y$ for $x = a, b$ and $y = a, b$

$xx \rightarrow s$ for $x = a, b$

$s's \rightarrow s$

Illustration:

abbababbab

$a'(b'(b'(a'b)))\ a'(b'(b'(a'b)))$

$s'(s'(s'(s's)))$

s

Example 3. The language $(a^n b^n)^m$

R' : $x\ x'y \rightarrow x'(x'y)$ for $x = a, b$ and for all $y \in S$

$ab \rightarrow t$

$t'x\ t'x \rightarrow t'x$ for all $x \in S$

$t \rightarrow s$

$s's \rightarrow s$

$T = \{s\}$

Illustration:

aaabbbaaabbb

$a'(a'b)\ b'(b'b)\ a'(a'a)\ b'(b'b)$

$t'(t't')\ t'(t't)$

$t'(t't)$

$s'(s's)$

s

Example 4. The language $a^m b^n c^{mn}$

R': $x\ x'y \rightarrow x'(b'y)$ for $x = b, c$ and all $y \in S$

$a\ b'x\ c'x \rightarrow b'x$ for all $x \in S$

$b \rightarrow s$

$s's \rightarrow s$

$T = \{s\}$

Illustration:

aaabbbbccccccccccc

$aaa\,b'(b'(b'b))\,c'(b'(b'b))\,c'(b'(b'b))\,c'(b'(b'b))$

$aa\,b'(b'(b'b))\,c'(b'(b'b))\,c'(b'(b'b))$

$b'(b'(b'b))$

$s'(s'(s's))$

s

Thus, the possibility to add further index levels at option provides a meansof performing arithmetical operations. The context-free multi-index rules are powerful and cover many languages of what is known as the context-sensitive type.

We shall now turn to linguistic interpretations of such a calculus.

Multi-index Calculus in Linguistics

The multi-index calculus can be applied in linguistics above all for two purposes: to replace context-sensitive rules and to provide a means of representing p-markers.

Context-free multi-index rules derived from context-sensitive rules

It is possible to replace many - all? - context-sensitive rules by an equivalent set of context-free multi-index rules.

Thus, the rule

$$a \rightarrow b / - c$$

can be replaced by

$a \rightarrow b'p$, $c \rightarrow c'q$ and $pq \rightarrow e$ or, more cautiously by the assignment rules

$$a \rightarrow A'p$$
$$c \rightarrow C'q$$

and the reduction rules

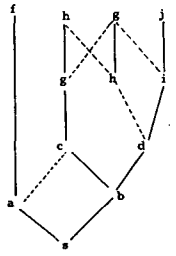
$$A'p \rightarrow A'r$$
$$A'r \rightarrow B'r$$
$$rq \rightarrow e$$
$$p \rightarrow e$$
$$q \rightarrow e$$

where e is the unity element.

Let us consider the following little grammar:

$$j \rightarrow i / g -$$
$$hg \rightarrow gh$$
$$i \rightarrow d / h -$$
$$gh \rightarrow c$$
$$f \rightarrow a / - c$$
$$cd \rightarrow b$$
$$ab \rightarrow s$$

With this grammar, the sentence fhgj will be analyzed thus



We have here adopted a "mixed" tree representation for context-sensitive structures, with obvious significance.

We can reduce the same sentence to s by the following set of rules:

j → i ' k

g → g ' l

lk → e

h → g ' m

g → g ' t

g → h ' n

mt → m

mn → e

i → d ' p

h → h ' q

qp → e

gh → c

f → a ' r

c → c ' t

rt → e

cd → b

ab → s

Thus,

f h g j

f h g'l i'k

f h gi'(lk)

f g'm h'n i

f gh'(mn) i

f g h'q d'p

f g hd'(qp)

f c d

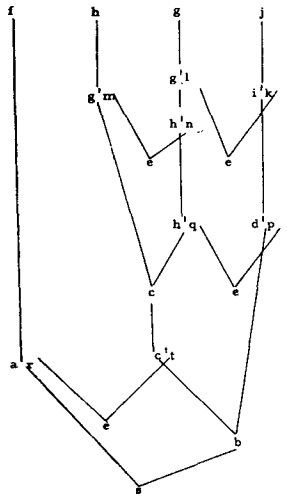
a'r c't d

ac'(rt) d

a b

s .

Graphically, this means that we have a set of interconnected treegraphs:



In a transformational grammar, we interpret G" as a grammar component, adding to our grammar a component $G' = < S', R', I', T' >$ where I' is the set T" of p-markers, T' is a subset thereof and R' is a set of multi-index rewriting rules such as

$$a'x \rightarrow a'y$$

$$a'x\ b'y \rightarrow c'x$$

$$a'x\ a'x\ b'y \rightarrow a'x\ b'y\ a'x\ b'y$$

$$a'x \cdot b'y \rightarrow b'y \cdot a'x$$

for specified sets of values for x, y, etc., that is, substitution, reduction, expansion and permutation rules for which the conditions are not confined to one index level at a time.

Regarding the analysis as a syntactic tree, we may characterize transformational rules as such where the conditions for some symbol(s) to be rewritten in a specified way refer to the "vertical" neighbours (not to the "horizontal" neighbours as in context-sensitive rules). We might speak about pretext and posttext sensitive rules, or generally about "kintext sensitive" rules. Obviously and notoriously, "kintext" must play a different role in generative and in recognition procedures, since pretext in one case is posttext in another.

Thus, one component may map the input strings on $T'' = \{ t_i'x \mid t_i \in T;\ x \in S'' \}$ and a transformation component may map $I' = T''$ on $T' = \{ t'y \mid t \in A \}$ and $y = \{ a_1'a_2'a_3' \ldots \mid a_i \in B \}$ where B is a subset of S" and $A \subseteq T$. Or we may define the target set for each component in other ways.

Multi-index calculus in a transformational grammar

Given a constituent structure grammar $G = < S, R, I, T >$ we obtain an infinite grammar G'' by replacing S by

$S'' = S \cup \{ s_1 's_2 's_3' \ldots \mid s_i \in S'' \}$ and R by

$$R'' = \{ a_i a_2 \ldots a_n \rightarrow b'(a_1 a_2 \ldots a_n) \mid (a_1 a_2 \ldots a_n \rightarrow b) \in R \}$$

if R is context-free and otherwise

$$R'' = \{ a_1 a_2 \ldots a_n \rightarrow b_1'(a_1 a_2 \ldots a_n) \cdot b_2'(a_1 a_2 \ldots a_n) \cdot \ldots \cdot b_m' (a_1 a_2 \ldots a_n) \mid (a_1 a_2 \ldots a_n \rightarrow b_1 b_2 \ldots b_m) \in R \}$$

and replacing $T = \{ t_1, t_2, \ldots, t_k \}$ by

$T'' = \{ t_i' x \mid t_i \in T \; x \in S'' \}$.

That is, we obtain a grammar* which maps given strings on an infinite set which may be considered as a set of p-markers**. G'' is then an interpretation grammar, corresponding to G.

* a decidable one, see p. 13 above, footnote. The number of levels does increase, but all rules refer exclusively to the uppermost level.

** These multi-index expressions naturally contain all information that transformations operate upon. Indeed, they will often contain too much, but superfluous indexes can easily be eliminated by multi-index rules; the point is that no side conditions for permissible transformational rewritings need be observed. Everything needed for the calculus is in the string.

Thus, one-level reduction rules suffice for a decision grammar for a constituent-structure language and multi-index reduction rules suffice for an interpretation grammar for such languages. Multi-index rules also suffice for a decision grammar for a transformationally defined language.* The question remains if they suffice for an interpretation grammar for the latter.

A structural description of the sentence may be given as the sequence of p-markers obtained during the analysis. Now, since the relative order of operations is not inherently fixed, we would like to find a representation of such sequences such that equivalence can easily be defined. That is, we want to find an adequate interpretative grammar corresponding to G'. Can multi-index rules serve those purposes ?

The unified formalization, provided by the multi-index representation, might prove an aid to finding an effective interpretative calculus for transformationally defined languages.

Conclusion

The multi-index calculus seems promising for several linguistic purposes, especially where restrictions can be assigned to several, weakly interacting levels.

* if this is decidable. They may also, incidentally, provide simple decidability criteria for a transformational grammar. Cf. the hints above (p. 13).