KYBERNETIKA ČÍSLO 3, ROČNÍK 1/1965

Analysis by Synthesis in the Light of Recent Developments in the Theory of Grammar*

G. H. MATTHEWS

The definition of the notion analysis-by-synthesis with special regard to natural language is done and its applications are discused.

This paper describes an algorithm for finding all and only the structural descriptions of any sentence with respect to a given grammar constructed in accordance with the theory of grammar which has been developed by linguists at the Massachusetts Institute of Technology – for the most part by N. Chomsky [1] – and which is discussed by E. S. Klima at this colloquium [2]. In an earlier paper [3], I described an analysis-by-synthesis algorithm for the class of grammars described by N. Chomsky in publications that had appeared prior to 1962 – especially his *Syntactic structures* [4]. In this paper, I assume a familiarity with the works [2], [3], and [4], restating parts of them only where it is necessary for clarity.

My work toward a useful and efficient sentence-analysis routine has been based on three assumptions:

a) only a routine which provides the complete structural description of any sentence with respect to the given grammar can be considered useful,

b) given the theory of grammar within which we have been working for the last several years, only routines which are – at least partially – synthetic can provide complete structural descriptions, and

c) given a grammar of a natural language there should be a general method – preferrably an algorithm – for constructing the sentence–analysis routine.

* Communicated at the *Prague Colloquium on Algebraic Linguistics and Machine Translation,* September 18th—22nd, 1964.

This work was supported in part by the U.S. Army, Navy, and Air Force under Contract DA 36-039-AMC-03200(E); in part by the National Science Foundation (Grant GP-2495), the National Institutes of Health (Grant MH-04737-04), the National Aeronautics and Space Administration (Grant NsG-496), and the U.S. Air Froce (ESD Contract AF19(628)-2487).

Given these three assumptions, it is natural to inquire into the status of my work in analysis by synthesis in the light of recent developments in the theory of grammar discussed in the works [1] and [2]. The quick answer to this inquiry is that – with a few minor changes – the original proposal presented in the paper [3] is still adequate. The second answer is that a considerably more useful and much more efficient analysis-by-synthesis routine can now be proposed which greatly reduces the need for preliminary analysis to – perhaps – merely a dictionary look-up routine. (You recall that the preliminary analysis was necessary in the earlier described routine in order to make it even possibly efficient enough for actual use.) What follows, then, is a sketch of an analysis-by-synthesis algorithm for the class of grammars described by what might cautiously be called the "current theory of grammar".

This analysis-by-synthesis method to be described here is based upon the fact that it is possible to assign to each sentence of a language a unique number which – as in my earlier paper – I refer to as the specifier of the sentence. Specifiers, themselves, are composed of a finite, but unbounded, sequence of vectors, each of which is unique to one of the kernel sentences of the language. In a given specifier, the order of these vectors determines just how each of these kernel sentences are related to one another within the sentence. The vectors are finite in length and they all have the same number of places in each of which can appear one of a finite number of entries. Furthermore, the number of possible entries for the i^{th} place of the vectors remains constant for all vectors and for all specifiers. Thus, there are a finite number of vectors that may be used to make up the specifiers of sentences.

To show the nature of the relationship between vectors and kernel sentences we will first consider a part of the phrase structure of English, i.e., as it appears to us today.

(1) $S \rightarrow \left\{\begin{matrix}(WH)\\ 1*1\end{matrix}\right\}\ (NEG)\ \ NP\ \ AUX\ \ VP\ \ (Post\text{-}S)/\ \left\{\begin{matrix}\#\\ 1*1\end{matrix}\right\}\ -$

(2) $VP \rightarrow (Asp)\ \ MV$

(3) $MV \rightarrow V\left\{\left\{\begin{matrix}\left\{\begin{matrix}COMP\\ PRT\\ Dir\end{matrix}\right\}\ (NP\ (AGNT))\\ ATT\ \ (NP)\ \ (AGNT)\\ Loc\\ (PRT)\ \ PRED\end{matrix}\right\}\ \ (RPS)\ \ (Man)\right\}\ \ (Frq)\ \ (Dur)$

(4) $Post\text{-}S \rightarrow (Loc)\ \ (Tim)\ \ (if\ COMP\ then)$

(5) $PRED \rightarrow \left\{\begin{matrix}ADJ\ \ (COMP)\ \ /(PRT)\ \ -\\ Loc\\ Det\ \ \ N\\ like\ \ \ NP\end{matrix}\right\}$

(6) $ATT \rightarrow \left\{ \begin{matrix} to \\ for \\ Prep \end{matrix} \right\} NP$

(7) $RSP \rightarrow Prep \quad NP$

(8) $Dir \rightarrow Prep \quad NP$

(9) $Man \rightarrow ADJ \quad Ly$

(10) $Frq \rightarrow Prep \quad NP$

(11) $Dur \rightarrow Prep \quad NP$

(12) $Loc \rightarrow Prep \quad NP$

(13) $Tim \rightarrow Prep \quad NP$

(14) $NP \rightarrow Det \quad N \quad ((COMP) \; / \left\{ \begin{matrix} - & AUX \\ V & - \end{matrix} \right\})$

(15) $Det \rightarrow (Pre\text{-}Art) \quad Art \quad Num \quad (Post\text{-}Art)$

(16) $NUM \rightarrow \left\{ \begin{matrix} Sg \\ Pl \end{matrix} \right\}$

(17) $Pre\text{-}Art \rightarrow (Deg) \quad \left\{ \begin{matrix} Integer \\ Degree \end{matrix} \right\} \quad of$

(18) $Post\text{-}Art \rightarrow (\left\{ \begin{matrix} \left\{ \begin{matrix} Integer \\ Total \end{matrix} \right\} \\ 1^{one} \quad 1 \end{matrix} \right\}) \quad (other) \; (Rel \; COMP) \; / \left\{ \begin{matrix} Pl \\ 1^{*}1 \end{matrix} \right\} \; -$

(19) $Art \rightarrow \left\{ \begin{matrix} Def \quad / \left\{ \begin{matrix} - & Pl & Total \\ (of) & - & \end{matrix} \right\} \\ Indef \end{matrix} \right\}$

(20) $AUX \rightarrow Tns \; (M)$

(21) $Asp \rightarrow (have \; En) \quad (be \; Ing)$

(22) $Tns \rightarrow \left\{ \begin{matrix} Pres \\ Past \end{matrix} \right\}$

(23) $AGNT \rightarrow be \quad En \quad by$

One important feature of this phrase structure is that it is a sequential grammar. A sequential grammar is one which for all i, if a symbol is expanded by rule (i), then there is no rule $(i + n)$ – for $n \geqq 0$ – which introduces that symbol. This means, of course, that the phrase structure generates a finite number of sentences. However, it does not mean that a rule cannot be applied several times within a single derivation. For example, by the time that rule (14) is to be applied, there can have been several

NP's introduced in the derivation. Rule (14) then has to be applied as often as possible, i.e., once for each *NP* in the current line of the derivation.

Many of these phrase structure rules are actually several rules of the type defined by Chomsky as phrase structure rules [5], which have been abbreviated by parentheses and braces. For example, rule (17) is an abbreviation for four different phrase structure rules that give expansions for the symbol *Pre-Art*, viz., *Deg Integer of*, *Deg Degree of*, *Integer of*, and *Degree of*. Rule (14) is an abbreviation for five phrase structure rules that expand the symbol *NP*. Two of these expand *NP* into *Det N COMP* and the other three expand it into *Det N*.* We refer to these several expansions of a symbol given by a rule as the subrules of that rule. Thus, each application of a rule requires a choice from among its applicable subrules. Finally, each rule in the grammar is obligatory, i.e., if the rule is applicable, it must be applied. We are now able to establish a useful relationship between vectors and kernel sentences. The positions in the vectors from left to right correspond to the rules of the phrase structure from the first rule to the last, and the possible entries in a given position correspond to the subrules of the rule associated with that position.

*) There is, in fact, an algorithm for unpacking parenthesized and bracketed rules into an ordered set of rules of the type defined by Chomsky, which also designates some of them as obligatory and others as optional. Rule (14) is an abbreviation for the subrules (14a) to (14e) where (14a) and (14b) are optional and (14c), (14d), and (14e) are obligatory.

(14a) *NP AUX → Det N COMP AUX*
(14b) *V NP → V Det N COMP*
(14c) *NP AUX → Det N AUX*
(14d) *V NP → V Det N*
(14e) *NP → Det N*

Similarly, rule (7) has one subrule, and rule (18) has eighteen subrules. To show how use is made of enumerated braces we give here these eighteen subrules of rule (18).

(18a) *PL Post-Art → Pl Integer other Rel COMP*
(18b) *Pl Post-Art → Pl Total other Rel COMP*
(18c) *Pl Post-Art → Pl Integer other*
(18d) *Pl Post-Art → Pl Total other*
(18e) *Pl Post-Art → Pl Integer Rel COMP*
(18f) *Pl Post-Art → Pl Total Rel COMP*
(18g) *Pl Post-Art → Pl Integer*
(18h) *Pl Post-Art → Pl Total*
(18i) *Pl Post-Art → Pl other Rel COMP*
(18j) *Pl Post-Art → Pl other*
(18k) *Pl Post-Art → Pl Rel COMP*
(18l) *Post-Art → one other Rel COMP*
(18m) *Post-Art → one other*
(18n) *Post-Art → one Rel COMP*
(18o) *Post-Art → one*
(18p) *Post-Art → other Rel COMP*
(18q) *Post-Art → other*
(18r) *Post-Art → Rel COMP*

Of these subrules, (18k) and (18r) are obligatory, the others are optional.

However, it is not quite as simple as this. One difficulty arises from the fact mentioned above that some rules may be applied several times because there may be several instances of the symbol that the rule expands in the current string of the derivation. In the particular grammar presented here, when rule (14) is about to be applied, there may be several *NP*'s in the string, due to the fact that *NP*'s are introduced by ten different rules, viz., rules (1), (3), (5), (6), (7), (8), (10), (11), (12), and (13). Furthermore, the symbol *Loc*, which is expanded by rule (12), may itself appear several times in the string when rule (12) is to be applied, for there are three rules that introduce *Loc*, viz., rules (3), (4), and (5). And, since the expansions of these several *NP*'s are independent from one another, there have to be twelve positions in the vector that govern rule (14), i.e., one position for each *NP* that might appear in a line of a derivation.* Similarly, the constituent *Det* is introduced by two rules — (5) and (14); but since rule (14) may be applied as many as twelve times in a derivation, we must assign thirteen positions in the vector to rule (15); and since this is the case with rule (15), it is also with rules (16) to (19). We should note here, however, that no matter how many instances of a symbol may be introduced by a phrase structure grammar, — and thus positions in the vector assigned to a single rule, — the restriction that the grammar be sequential assures us of vectors of finite length.

We stated earlier that every rule of the phrase structure grammar is obligatory. Consider now rules (7), (8), (9), (10), (11), (12), (13), and (23). Each of these rules has just one subrule; so, since every rule is obligatory for as long as it is applicable, these rules in fact present no choice. How each of these rules is applied is determined entirely by the rule itself, and whether one of these rules is to be applied is entirely determined by the choice of subrules made earlier in the grammar. This means that there is no need to have positions in the vector corresponding to these rules. In this particular grammar, the statement of rule (14) is such that it is easy to see that only in the case of the subject *NP* and some instances of the object *NP* is there a choice. in the expansion of *NP*; in all other cases, *NP* must be expanded into *Det N*. We still need a position in the vector for the object *NP* for those cases when we do have a choice Furthermore, our algorithm would also provide positions for the other *NP*'s, for in

* For this phrase structure grammar, we actually need to provide only eight positions for rule (14). This is because there are many combinations of applications of rules that cannot be used in the derivation of a sentence. In the case of *NP*'s, rule (1) introduces one, rule (3) may introduce an *NP* and at most four other constituents which, by rules (6), (7), (8), (10), and (11) in turn introduce *NP*'s, and finally rule (4) introduces a *Loc* and a *Tim* which by rules (12) and (13) provide for two more *NP*'s. Thus, a line of a derivation cannot contain more than eight *NP*'s. However, in keeping with the goal of providing an algorithm that — given any grammar — automatically gives an analysis routine for the structural descriptions of sentences with respect to that grammar, we will not make use of any information which cannot be obtained by any but the most cursory examination of the grammar. Data of this sort, arising from a study of the grammar as such, is of value for increasing the efficiency of an analysis routine which has already been given by the algorithm.

general it is not possible in principle to know what the environments of any given symbol may be in an arbitrary context-sensitive grammar.

We have now seen a need for a vector, or equivalently, a program which will govern the operation of this 23-rule phrase structure grammar. Due to the fact that it must have several positions corresponding to some of the rules, and no position corresponding to others, as well as the possibility of having an algorithm which will define its structure given the grammar; we have seen the need for the vector for this grammar to have 86 positions. Perhaps now it will clarify what we have said if we consider a concrete example. We will consider a vector, and for convenience we will place over each position the symbol which is expanded by the corresponding rule.

S	*VP*	*MV*	*Post-S*	*PRED*	*ATT*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*
(8	1	192	0	0	0	3	0	0	0	0	0	0	0	0	0	0

Det	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*Det*	*NUM*	*NUM*	*NUM*
4	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0

NUM	*NUM*	*NUM*	*NUM*	*NUM*	*NUM*	*NUM*	*NUM*	*NUM*	*NUM*	*Pre-Art*	*Pre-Art*
0	0	0	0	0	0	0	0	0	0	0	0

Pre-Art	*Pre-Art*	*Pre-Art*	*Pre-Art*	*Pre-Art*	*Pre-Art*	*Pre-Art*	*Pre-Art*	*Pre-Art*	*Pre-Art*
0	0	0	0	0	0	0	0	0	0

Pre-Art	*Post-Art*	*Post-Art*	*Post-Art*	*Post-Art*	*Post-Art*	*Post-Art*	*Post-Art*	*Post-Art*
0	0	0	0	0	0	0	0	0

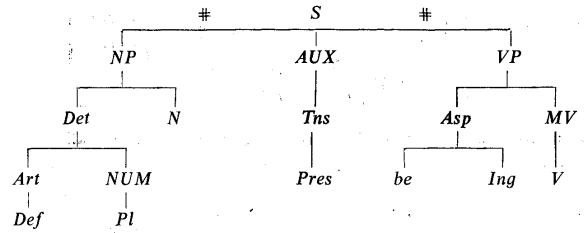
Post-Art	*Post-Art*	*Post-Art*	*Post-Art*	*Post-Art*	*Art*	*Art*	*Art*	*Art*	*Art*	*Art*	*Art*	*Art*
0	0	0	0	0	3	0	0	0	0	0	0	0

Art	*Art*	*Art*	*Art*	*Art*	*AUX*	*Asp*	*Tns*
0	0	0	0	0	2	3	1)

This vector specifies that the following subrules of the phrase structure grammar are to be applied.

S	8	$S \rightarrow NP \quad AUX \quad VP$
VP	1	$VP \rightarrow Asp \quad MV$
MV	192	$MV \rightarrow V$
NP	3	$NP \quad AUX \rightarrow Det \quad N \quad AUX$
Det	4	$Det \rightarrow Art \quad NUM$
NUM	2	$NUM \rightarrow Pl$
Art	3	$Art \rightarrow Def$
AUX	2	$AUX \rightarrow Tns$
Asp	3	$Asp \rightarrow be \quad Ing$
Tns	1	$Tns \rightarrow Pres$

And this set of subrules generates the following constituent structure:



We haven't introduced lexical items yet, but when that is done, this constituent structure will underlie such sentences as *The leaves are burning*. If we consider the verb stems that might appear in this structure, we see that they are just the intransitive verbs that can have a noun – as distinct from a nominalized sentence – as the subject. But this fact, i.e., that an intransitive verb that may have a noun as its subject, must be used in this structure, does not represent any choice made at this point in the derivation, rather it is completely determined by the choices of subrules made in rules (3), (5), and in the expansion of the subject *NP* in rule (14). In fact, the class of verbs from which a stem must be chosen is always determined by the subrules that were chosen in these three rules. We can even say then that the name of the verb classes are the ordered sets of subrule choices that may be made from rules (3), (5), and the expansions of subject and object *NP*'s from rule (14). Thus, the name of the verb class whose members may appear in the structure above is (192, 0, 3, 0). Similarly, the class of nouns is determined by the choice of subrules made from rules (15) to (19). Thus, in this sentence the noun is chosen from class (4, 2, 0, 0, 3). In addition to the verb and the nouns, we also have several other terminal symbols in this phrase structure which must be expanded into specific stems. These are *PRT* introduced by rule (3); *ADJ* introduced by rules (5) and (9); *Prep* introduced by rules (7), (8), (10), (11), (12), and (13); *Deg* and *Degree* introduced by rule (17); *Integer* introduced by rules (17) and (18); and *Total* introduced by rule (18). Assuming again that these are all completely independent of one another – which of course is not the case – we must add 76 more positions to the vector, for each of these stem classes contains several stems among which to choose from in order to generate a complete sentence.

Up to now we have systematically ignored the symbol *COMP* which is introduced by several rules of the phrase structure. We now add one more phrase structure rule to this grammar which will make the grammar what I call a cyclical sequential grammar.

$$(24)\qquad COMP \rightarrow \#\left\{\begin{array}{l} Past\ \ will \\ for\ \ to \\ WH \\ \left\{\begin{array}{l} that \\ Poss\ \left\{\begin{array}{l} NOM\ \ of \\ Ing \end{array}\right\} \end{array}\right\}1 \end{array}\right\} S\ \ \# \ /\left\{\begin{array}{l} if \\ (ADJ) \\ (Rel) \\ 1*1 \end{array}\right\} -$$

This rule reintroduces the initial symbol of the grammar, and thus allows for an infinity of sentences to be generated by the phrase structure. There is just this one rule which has this property of introducing a symbol that is expanded by an earlier rule of the grammar; and this earlier rule is in fact the first rule of the grammar. Another important characteristic of rule (24) as that the symbol which is expanded thereby is what might be called a generalized modifier constituent. The rules that introduce it – rules (3), (4), (5), (14), and (18) – use it as a modifier of some other constituent that is introduced by the same rule. So what this phrase structure is saying is that all modifier constituents in a natural language are primarily sentences of that language. Note incidentally that rule (1) states that an *S* dominated by *COMP* cannot be expanded into a sequence beginning with *WH*.

Now with this type of phrase structure, it is clear that what we have actually done is to generate the underlying structure for all the sentences of the language directly within the phrase structure part of the grammar. What remains for the transformational component of the grammar is to suitably modify the actual form of the several subtrees of the derivation which are dominated by the initial symbol *S*. In fact, some of the elements that are used in this modification are themselves introduced by rule (24). For example, when the symbols *Past will* are inserted into the following clause, the result is a conditional clause of the type usually beginning with the conjunction *if*. A *COMP* that modifies an adjective must have the two morphemes *for* and *to* added to it. Relative clauses must begin with the element *WH*. Finally, *for to* clauses, *WH* clauses, *that* clauses, and certain nominalizations may appear as parts of *NP*'s. Rule (24), of course, does not exhaust all of the types of dependent constructions of English; but it is in the general format of the rule that does so. We must now add one more position to the vector to accomodate rule (24).

This completes the description of the vectors for kernel structures. From here it is a simple step to the description of specifiers. As we said above, a specifier is a finite but unbounded sequence of vectors. In the computation of a vector, the application of rule (24) as often as it applies introduces a certain number of the symbol *S*. For each one of these *S*'s introduced, we must have a vector to govern its expansion. Thus, if we order the several vectors of a specifier in a Polish-like order, the number of *S*'s generated by each vector will be the number of vectors that it immediately dominates. Thus, a properly constructed specifier is a linear representation of the complete underlying structure of some sentence.

Note now that the phrase structure introduces various elements which have the

effect of naming a transformation which must be applied in order to transform the underlying structure of a sentence into its surface structure, i.e., into the structure and shape it has on the printed page. For example, rule (1) may introduce the element *WH*; the presence of this element indicates that the question transformation must be applied to the kernel structure within which this appears. Rule (3) may introduce the element *AGNT*, which calls for the application of the passive transformation, the characteristic morphemes of which are introduced by rule (23). Similarly, what now corresponds to generalized transformations are also obligatorily applied; the elements introduced by rule (24) designate which of these transformations must be applied in order that the kernel structure dominated by the *S* take on the proper shape of the type of dependent clause it is and also be properly positioned with respect to the other clauses of the sentece.

In my previous publication on analysis by synthesis [3], I discussed how specifiers of sentences might be used in an analysis program for sentences of a natural language. The specifiers described here assume no previous use of a preliminary analysis program. However, the simplest type of analysis, viz., a dictionary that assigns one or more parts of speech to each word, would greatly increase the efficiency of an analysis-by-synthesis program: It would allow the removal of that part of a vector which makes a choice of a stem from a class. This alone reduces the number of positions in the vectors for the grammar we have presented here from 163 to 87, and in addition, these positions so removed are just the ones which present the greatest number of choices. Further reduction in the length of the specifier can sometimes be made through study of the grammar itself. For example, in the grammar here, rule (14) presents a choice of subrules only in the case of the *NP*'s that are introduced by rules (1) and (3); thus the vector need not have positions assigned for the control of the expansions of the other *NP*'s that might appear in the derivation anymore than it needs positions to govern the expansion of such symbols as *Loc* and *Man*. This reduces the vector to 77 positions.

One important characteristic that specifiers as described in this paper as distinct from those described earlier is that there is a direct relationship between the vectors and the kernel structures of the language as well as between the entries in a given position and the way that a specific symbol is expanded. This should make it quite easy to define operators which would, for example, convert the specifier for any sentence of a class into one for a paraphrase thereof. Operators might also be definable for converting specifiers into routines which search through a previously stored and structured body of data. We might also look forward to a time when sufficient knowledge of semantics has been acquired when operators might be definable to convert the specifiers of sentences of one language into those of their translations in another.

(Received November 9th, 1964.)

[1] Chomsky N.: Aspects of a Theory of Syntax. M.I.T. Press (to appear).
[2] Klima E. S.: Current Developements in Generative Grammar. Kybernetika 1 (1965), No. 2, pp. 184—197.
[3] Matthews G. H.: Analysis by Synthesis of Sentences of Natural Languages. In Proceedings of the 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, National Physical Laboratory, Symposium No. 13, pp. 531—542, London 1962.
[4] Chomsky N.: Syntactic Structures. The Hague 1957.
[5] Chomsky N.: On Certain Formal Properties of Grammars. Information and Control 2 (1959), pp. 137—167.

VÝTAH

Analýza vět syntézou ve světle současného vývoje teorie gramatiky

G. H. MATTHEWS

V prvé části stati se shrnují dosavadní výsledky zkoumání v oblasti teorie gramatiky. Nejdůležitější je tu zavedení odlišování derivované a původní struktury věty. Zjištění této distinkce a její přímé důsledky umožňují značně zjednodušit formulaci analýzy vět přirozených jazyků pomocí syntézy. Tato část rovněž obsahuje explicitní definici pojmu *analýza syntézou*, a to v podobě vhodné pro práci s přirozenými jazyky.

Druhá část obsahuje podrobný popis obecného programu analýzy syntézou a několik doplňkových postupů, které zvyšují její účinnost.

V poslední části práce jsou rozebrány některé možné aplikace obecného programu popsaného v části druhé.

G. H. Matthews, Department of Modern Languages and Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts, U.S.A.