[*International Conference on Machine Translation of Languages and Applied Language Analysis*, National Physical Laboratory, Teddington, UK, 5-8 September 1961]

ANALYSIS BY SYNTHESIS OF SENTENCES OF NATURAL LANGUAGES*

by

G. H. MATTHEWS
(Centre for Communication Sciences
R.L.E.
Massachusetts Institute of Technology)

IN several previous papers, I have described a mechanical fail-safe sentence-recognition routine that, in principle, cannot fail to give all and only the possible syntactic analyses for any string of symbols with respect to a given grammar. The first part of this paper offers a slightly more precise formulation of this recognition routine. In the second part of this paper I describe some subroutines that make possible a practical analysis-by-synthesis computer programme. In the last part I discuss some advantages of this type of recognition routine over others that have been proposed.

The analysis by synthesis of sentences is based upon the structural properties of grammars of natural languages. A grammar consists of a finite, though rather large, set of sentence-formation rules, i.e., grammatical rules. These rules provide for a denumerably infinite set of sentences, all of which draw their symbols from a single finite source, i.e., vocabulary. There are two kinds of vocabulary: Nonterminal vocabulary is found only in intermediate strings which occur during the derivation of a sentence. Only terminal vocabulary is found in sentences. (The term "string" refers to any sequence of terminal and/or nonterminal vocabulary, including sentences.)

Another characteristic of grammars is that the grammatical rules are applied in a given linear order, and each rule transforms that string which is the result of all previous rules in the ordering. A rule consists of a finite set of subrules, one of which is applied at each application of the rule. The set of subrules that has been applied in the derivation of a string is essentially the syntactic structure of

* This work was supported in part by the National Science Foundation, and in part by the U.S. Array (Signal .Corps), the U.S. Air Force (Office of Scientific Research, Air Research and Development Command), and the U.S. Navy (Office of Naval Research).

that string. For a rule to be applicable to a string, the string must be analyzed in terms of its structure in a way specified by the rule. If the rule is applicable, the string is transformed into another string in a way specified by one of the subrules. No rule can be applied if it is not applicable, and every rule is applied only at its place in the ordering. There are two kinds of rules. Optional rules do not need to be applied even though they may be applicable. Obligatory rules must be applied if they are applicable. All rules are recursive, that is, they may be reapplied any number of times as long as they are applicable, and, if they are obligatory, they must be reapplied until they are no longer applicable. There is no chance that, in any given case, there will not be a limit to the number of times that some obligatory rule is applicable. The rules of a grammar can be formulated in such a way that every possible combination of subrules conforming to the restrictions just described produces a different sentence of the language. Since there is no limit to the number of times that some of the optional rules may be applied in the production of sentences, there is no limit to the number of sentences of a natural language.

Each time a given rule is applied in the derivation of a sentence, a different one of its subrules can be used. For each rule of the grammar we order all possible permutations, with repetitions, of its subrules in such a way that, for any m, all permutations of m subrules precede all those of m + 1 subrules. And for each rule we map these permutations onto the integers. Consider the set of base-infinity numbers, all of which have n places, $r_1\ r_2\ \ldots\ r_n$, where n is the number of rules of some grammar. We associate with each place r_k the k^{th} rule in the ordering of the grammatical rules, and we associate with each integer in place r_k that permutation of subrules of rule k that was mapped onto that integer as we have just described. There is now a one-to-one correspondence between the numbers $r_1\ r_2\ \ldots\ r_n$ and the possible permutations of all of the subrules of the grammar, whether they are applicable or not. And among these permutations there will be those that produce the sentences of the language. Thus, the sentences of a natural language are denumerable. The types of permutations that correspond to some of the numbers $r_1\ r_2\ \ldots\ r_n$ and do not produce sentences, are those that produce strings of the language containing nonterminal vocabulary (we shall call these "nonterminated" derivations), and those that indicate that some rule which is not applicable is to be applied (we shall call these "blocked" derivations). Hereafter, I shall refer to the number that corresponds to a particular derivation as the "specifier" of that derivation.

We shall now make a closer examination of grammatical rules. All subrules have the effect of replacing a string of symbols by another non-identical string of symbols, i.e., the string $A_1\ A_2\ \ldots\ A_n$ is replaced by

B_1 B_2 ... B_m, where both n and m are greater than 0, and n # m and/or, for some k, $A_k \neq B_k$. For most subrules of grammars $n \leq m$, but for a relatively small number of subrules (to which I shall refer as ellipsis rules), n = m + 1. Each ellipsis rule deletes a single symbol when that symbol occurs in a specific environment, and changes that environment in such a way that it cannot be acted upon either as the deletable symbol or as the governing environment of any ellipsis rule.

Thus, in general, the number of symbols in a string increases with the number of subrules applied in its derivation. These restrictions on the application of the ellipsis rules makes it possible to define two functions: Given the number of symbols p in a string, we can calculate the largest number of rule applications m(p) from which a string of p symbols can be derived; and we can also calculate the number of symbols f(p) in the longest possible intermediate string of this string's derivation.

We now order the numbers r_1 r_2 ... r_n, i.e., the specifiers, in such a way that, for any m, all numbers the sum of whose places is equal to m precede all those for which this sum equals m + 1. We shall refer to this sum as the "specifier-sum". Specifiers with the same specifier-sum are ordered numerically. If we now assign the same order to the derivations that correspond to these numbers, it means that derivations containing a smaller number of rule applications will, in general precede those containing a larger number of rule applications. In fact, it is possible to define a function such that given the number of rule applications a, we can calculate the largest specifier-sum s(a) such that some derivation whose specifier has that specifier-sum contains a rule applications. We can now define a procedure by which, given any string of symbols, we can decide whether that string is a sentence of some given language. (1) Count the number of symbols in the given string (referred to hereafter, as the input string). Let this number be p. (2) Calculate s(m(f(p))) + 1. (3) Derive all sentences in the order described in this paragraph until the specifier of the next derivation to be carried out has a specifier-sum equal to the number calculated in step (2). (4) Compare each derived sentence (ignoring nonterminated and blocked derivations) with the input string, and if any of them matches it symbol-for-symbol, then the input string is a sentence of the language, otherwise it is not. Furthermore, for every possible different derivation, that is, for every different syntactic analysis, of the input string, this procedure will produce a sentence that matches the input string symbol-for-symbol. And, because of the particular relationship between the specifiers of the sentences and the grammar of the language, each specifier that guides the derivation of a matching sentence is an unambiguous representation of one of the possible syntactic analyses of the input string.

We have now shown that, in principle, it is possible to provide all of the possible syntactic analyses (with respect to a given grammar) for any string of symbols by means of a mechanical analysis-by-synthesis procedure. When one considers, however, the enormous number of sentences of natural languages of only average length, one wonders about the practicality of such a recognition procedure. Consider, for example, a language of 10,000 words (a rather conservative figure for any of the European languages); the number of different strings of n words, or less, would be $10^{4(n+1)}$ However, not all of these are sentences; but if we take into account the 50% redundancy of English calculated by C.E. Shannon[3], we can give $10^{2(n+1)}$ as a reasonable estimate for the number of sentences. Suppose, now, we have a recognition routine for this language which is carried out in the same way as the decision procedure described above, and we wish to determine all of the possible syntactic analyses for some string that is twenty words long (the average length of an English sentence). This recognition routine would have to derive and examine all of the sentences that are twenty words long, or shorter, i.e. 10^{42} sentences, as well as many that are longer, not to mention the much larger number of nonterminated and blocked derivations. Compare this number with the estimated number of seconds that have elapsed since the creation of the Earth, approximately 3.10^{17}, or the number of centimeters from here to the furthest known star, 2.10^{24}. It is quite clear that such a recognition routine is impossibly impractical for any existing or envisioned computer, or even battery of computers. However, there are short cuts that can be used in such a recognition routine which greatly reduce the number of sentences that must actually be derived, as well as the length of the derivations themselves. It is possible to delimit, for each input string, a noninitial part of the ordering of the derivations such that all possible derivations of that string occur within this noninitial part. Secondly by means of procedures called "preliminary analysis" and "feedback", we can, in the course of the analysis-by-synthesis routine for each input string, progressively narrow the set of derivations that must be considered as possible analyses. And, finally, the actual carrying out of the derivations can be considerably shortened.

The first type of short cut has the effect of eliminating from consideration a large number of the strings that are shorter than the input string without actually deriving these shorter strings. This is done by not beginning step 3 of the decision procedure at the beginning of the ordering but rather at some later point which, however, still precedes the input string. Earlier in this paper, we established an ordering of the specifiers such that the specifier-sums increase numerically. In addition, we set up a correspondence between the number in each place of the specifiers and the number of applications of the rule corresponding to that place such that these two numbers increase

together. We also saw that the grammatical rules are structured in such a way that, in general, the length of sentences increases with the number of rule applications. Because of this particular relationship between the specifier-sums, the number of rule applications, and the length of strings, it is possible to define a function - specific to each language - which calculates the smallest specifier-sum such that some specifier with that specifier-sum corresponds to a string with the same number of symbols as the input string. With such a function, in step 3 of the decision procedure we need not begin the derivations at the beginning of the specifier ordering: instead, by starting with specifiers of the minimum specifier-sum that corresponds to a string with the length of the input string, we can skip over many of the derivations that produce strings that are shorter than the input string.

For any recognition routine that is actually in use, it would probably be more economical to compile a table which, for each input-string length, would give the first and last specifiers of the ordering that correspond to terminated sentences of that length. This table could be compiled in conjunction with the actual operation of the routine. Thus, the short cut just described would be used only once for each input-string length and thereafter use of the table would replace the need for counting the number of symbols in each derived string. Such a table, of course, would have to be constructed for each language. We shall refer to the range of derivations defined for a given input string by this table as the "neighbourhood" of that string.

There is no question that the method just described eliminates from consideration a large number of derivations. However, there still remains an unwieldy number of derivations within the neighbourhood of any input string. Thus other short cuts are necessary. The next short cuts to be described make use of the notion "linguistically significant class" of derivations. A class of derivations is linguistically significant if, in deriving the members of this class, the same permutation of subrules of one and the same grammatical rule is applied in all of them. This is a generalization of the intuitive notion, "sentence type". The specifiers for the members of a linguistically significant class are all alike in that the same integer occurs in the place that corresponds to the particular defining rule. In addition, the logical product of two linguistically significant classes, if it is nonempty, is also a linguistically significant class, and the specifiers for such a class will have more than one place containing the same integer. Note that if the product of two classes is empty, it is because each class is defined by a different permutation of the same rule; the corresponding class of specifiers would be a class in which all of the members have simultaneously two different integers in the same place - an impossibility. Thus, the system of specifiers provides a convenient notation for designating a class

of derivations: A specifier that has only some of its places uniquely specified and the others left free designates a class of derivations. If all of its places are uniquely specified, then this class consists of just one derivation. The following procedures will produce a sequence of linguistically-slgnificant-class designations connected to each other by the logical connectives *or* and *and not,* and parentheses. A possible sequence might be: (a $\vee$ b) $\wedge$ -c, in which a, b, and c are partially specified specifiers. This would indicate a set of sentences composed of those members of the union of the linguistically significant classes a and b that are not also members of the linguistically significant class c.

The first of these procedures that use the notion "linguistically-significant class" is "preliminary analysis". This procedure examines the input string directly, searching for features that would place it in one or more linguistically significant classes of derivations - if it is, in fact, a derivation of the language - and/or exclude it from one or more classes. Then the recognition routine need not produce any derivations that do not conform to these class-membership restrictions. It is clear that the more sophisticated the preliminary analysis is, the more narrowly defined is the class, or classes, to which the input string can be assigned. In fact, most of the mechanical-translation and information-retrieval projects have attempted to develop a recognition routine that is essentially equivalent to preliminary analysis as described here; but one whose output is those fully specified specifiers that correspond to just the correct derivations of the input string. Note, however, that the success of analysis-by-synthesis is not dependent upon such a complete preliminary analysis; instead it is a method for reducing the number of operations that must be performed in order to analyze a sentence.

A failing derivation, that is, one that does not produce a string that matches the input string, can be used to define more narrowly the set of derivations that includes those which do produce the input string. This feedback is possible because if the derivation does not produce a matching string, this fact can in the great majority of cases be determined before the derivation is complete. A partially complete derivation is equivalent to a linguistically significant class of derivations, for the initial part of its specifier is uniquely specified and the rest is unspecified. Thus, each failing derivation will define a class of derivations to which the input string does not belong. A class of derivations defined in this way will have had only one of its members considered and even this one will have been only partially derived. Furthermore, this class will overlap with the set of derivations defined by the neighbourhood, the preliminary analysis, and the feedback already carried out, because one of its member derivations, was, in fact, considered.

There are several ways of determining that a derivation will fail before it is completed. If a derivation is blocked, it fails because it

cannot be completed. When a terminal symbol is introduced into the derivation, it must be one of those that occurs in the input string. If it is not, the derivation will fail. The neighbourhood of an input string contains derivations that produce strings that are longer than the input string. At some point in the derivation of such a string, the current intermediate string will have enough symbols so that even if all of the applicable ellipsis rules that follow in the grammar were to be applied, the final string would still be too long; thus this derivation will fail.

Earlier in this paper we established an ordering of the specifiers based on their specifier-sums, and those specifiers with the same specifier-sum we placed in numerical order. One of the reasons for this particular ordering is that each specifier can give rise to the next one in the ordering by a simple modified addition function. Another reason is that with this ordering plus the structure of the specifiers themselves the number of operations that are necessary to carry out a sequence of derivations can be considerably cut down. The specifiers are constructed so that the order from left to right of the places in the specifiers corresponds to the order of the rules in the grammar. Thus, since the specifiers are in numerical order, some initial part of derivations that follow each other in the ordering will be the same, and this initial part can be preserved and used in the next derivation without actually having to be carried out again.

We have now seen that a recognition routine based upon the decision procedure for sentences of a natural language can be made into a practical computer programme by use of the several short cuts described above. (I do not mean to imply that these are the only possible short cuts that could be programmed into this recognition routine.) The question now to be considered is whether there are any advantages in such a recognition routine over those routines more commonly known, which are such as to justify the research into the basic structure of natural and artificial languages that has been and will continue to be necessary, as well as the additional effort needed to convert the results of this research into a computer programme. One important aspect of this routine is its treatment of input strings that are not sentences. This recognition routine is fail-safe, in that if a string cannot be derived from a given grammar, then the recognition routine will indicate this fact - it will not give a false analysis. This is because no derived terminated sentences will match the input string. However, in the process of going through the recognition routine, preliminary analysis and feedback will have defined in terms of partially specified specifiers, a set of derivations which will indicate a great deal about the input string. We have seen that a partially specified specifier is equivalent to a partially defined derivation. However, a partially defined derivation is a string with some of its syntactic structure indicated and the rest unknown. Thus, wherever the partially

specified specifiers that are produced by preliminary analysis and feedback are in agreement about which rules to apply, there we can say something definite about the syntactic structure of an input string that is not a sentence.

There are certain other advantages to this recognition routine. It is in no way dependent upon the structure of any particular language, for the grammar of the input language is just one of the subroutines of the whole recognition routine, and it is a self-contained subroutine that has no effect upon the structure of the others. Furthermore, in any mechanical-translation scheme that separates the recognition of the syntactic structure of input sentences from the construction of their translations, the grammar in this type of recognition routine is exactly the same as that of the output language in the sentence-construction routine of a mechanical-translation programme which translates into this language. This means that if a grammar of some language is written for mechanical translation, it will serve in any mechanical-translation scheme that translates either into or out of this language. Finally, the grammars that are necessary for this recognition routine have the same formal properties as those that are being written by linguists in fields other than mechanical translation, and thus the work of many people outside of the mechanical-translation field can be used directly in mechanical-translation programmes.

REFERENCES

1. MATTHEWS, G.H. Denumerability of the Sentences of Natural Languages, American Mathematical Society Meeting, New York, April 1960.
   --- Recognition of the Structure of Sentences. Quarterly *Progress Report No. 58,* Research Laboratory of Electronics, Massachusetts Institute of Technology, 1960, pp. 224-227.
   --- Use of Grammars within the Mechanical Translation Routine. *Proceedings of the National Symposium on Machine Translation* (to appear).
2. CHOMSKY, N. Logical Structure of Linguistic Theory, 1955. (unpublished, mimeographed; microfilm available at M.I.T. library).
   --- Note on Phrase Structure Grammars. *Information and Control* 1959, **2**,393-395.
   --- On Certain Formal Properties of Grammars. *Information and Control* 1959, **2**, 137-67.
   --- On the Notion "Rule of Grammar". *Proceedings of the Symposia on Applied Mathematics,* **XII** (to be published by the American Mathematical Society)
   --- Syntactic Structures. The Hague; Mouton and Company 1957
   --- and MILLER G.A., Finite State Languages. *Information and Control,* 1958, **1**, 91-112.

3. SHANNON, C.E. Mathematical Theory of Communication. *Bell Syst. tech J.* 1948, **27**, 379-423; 623-56
--- Prediction and Entropy of Printed English. *Bell Syst. tech. J.* 1951, **30**, 50-64.