[*Automatic Translation of Languages*: NATO Summer School…July 1962]

ON THE EQUIVALENCE OF MODELS OF LANGUAGE USED IN THE FIELDS OF MECHANICAL TRANSLATION AND INFORMATION RETRIEVAL*

MAURICE GROSS
Massachusetts Institute of Technology†

Summary—The purpose of this paper is to compare a certain number of well known models used in the fields of Mechanical Translation (M.T.) and Information Retrieval (I.R.). Different surveys of this type exist (Bar-Hillel [1], Hays [2], Lecerf [3], Sestier and Dupuis [4]), where models have been compared from the point of view of practical and linguistic adequacy. We wish here to compare certain formal characteristics of these models, in fact to show that they are strictly equivalent to a well studied model. The notion of equivalence will be defined formally, the model to which the other models are equivalent is Chomsky's model of context-free languages (c.f. languages).

The equivalences discussed here have not only an abstract character; several practical problems which arise naturally are clarified.

1

THE majority of MT and IR projects have been primarily concerned with the construction of grammars and computer programs aiming to produce syntactic analyses for the sentences of a given natural language.

In certain cases, the grammar and the recognition routine are completely amalgamated into a single program and the grammatical information is no longer available for a program that would synthesize sentences. The interest of synthesizing sentences has been shown, synthesis by the output of a transfer grammar, Yngve [5], or synthesis at random, Yngve [6]. This sort of grammar cannot pretend to hold for models of languages, unless one admits that human beings use alternatively two disjoint devices; one for reading, the other for writing; and probably two others, one for hearing, the other for speaking. One would have to construct four types of corresponding devices for each language in order to translate and to retrieve information automatically.

Rather frequent are the cases where the grammar is neutral between the programs of analysis and synthesis; these grammars following Chomsky [7] [8], we will call generative grammars.

We shall now make more precise the concepts of grammar and language, and examine the requirements a grammar has to meet [9].

We consider the finite set $V = [a_i \mid 0 \leq i \leq D]$ V is called vocabulary, the a_i's are words, a_0 is the null word. On V the operation of concatenation defines the set $C(V)$ of strings on V, any finite sequence of words $T = a_{i1}\ a_{i2}\ .\ .\ .\ a_{ik}$ with $0 \leq i_1\ i_2\ .\ .\ .\ i_k \leq D$ is a string on V, $C(V)$ is a free monoid whose generators are the a_i's.

* Presented at the NATO Advanced Study Institute on Automatic Translation of Languages, Venice, 15-31 July 1962.

† Presently at the "Institut Blaise Pascal—C.N.R.S". Paris, France.

(1) A subset L of $C(V)$ is called a language on $V : L \subset C(V)$.
(2) A string S, such that $S \in L$, is a sentence of the language L.
(3) A finite set of finite rules which characterize all and only the sentences of L, is called a grammar of L. (Productions of a combinatorial system: Davis [32].)
(4) Two grammars are equivalent if they characterize the same language L.

This abstract and most general model is justified empirically as follows:

The words (or morphemes) of a natural language L form a finite set, i.e. the vocabulary or lexicon V of the language L.

Certain strings on V are clearly understood by speakers of L as sentences of L, others are clearly recognized as nonsentences, so L is a proper subset of $C(V)$.

Many facts show that natural languages have to be considered as infinite. The linguistic operation of conjunction can be repeated indefinitely, the same thing for the embedding of relative clauses as in the sentence: {*the rat* [*the cat* (*the dog chased*) *killed*]*ate the malt*]. Many other devices of nesting and embedding exist in all natural languages and there is no linguistic motivation which would allow a limit on the number of possible recursions.

We wish to construct a grammar for a natural language L, that considered abstractly enumerates (generates) the sentences of the language L, and associated with a recognition routine it recognizes effectively the sentences of L.

The device so far described is a normative device which simply tells whether or not a string on V is a sentence of L; equivalently, it is the characteristic function of L. This minimum requirement of separating sentences from non sentences is a main step in our construction. The construction of this normative grammar requires the use of the total grammatical information inherent in L, and under these conditions it should be natural to use this same information in order to give as a by-product a description of the organization of a sentence S_1 in L. This can be done simply by keeping track of the grammar rules that have been involved in the analysis of S_1. This particular ordered set of rules is called the derivation of S_1. This by-product is of primary importance; it is a model for the understanding of L by its speakers (Chomsky [9], Tesnière [10]); parallelly any MT or IR realization requires the machine to have a deep understanding of the texts to be processed. Furthermore a normative device would not tell whether a sentence is ambiguous or not; the only way to describe the different interpretations assigned to an ambiguous sentence is to give them different descriptions.

The basic requirements for a formalized description of natural languages, almost trivial in the sense that they make practically no restrictions on the forms of grammars and languages, do not seem to have been widely recognized in the fields of MT and IR, nevertheless they are always unconsciously accepted.

Chomsky [11, 8, 12] has studied a large variety of linguistic and formal constraints that one can reasonably put on the structure of grammars. The grammars so constrained range from finite-state grammars to the device described above, which can be viewed as arbitrary Turing Machine.

These grammars in general meet a supplementary requirement; each derivation of a sentence has to provide a particular type of structural description which takes the form of simple trees or, equivalently of parenthesized expressions; both subtrees and parentheses are labelled, i.e. carry grammatical information. Certain types of grammar have been proved inadequate because of their inability to provide a structural description for the sentences they characterize (type 1 grammars in Chomsky [8]).

Many authors in the field of MT and IR start from the postulates:

(1) A grammar is to be put, together with a recognition routine, into the memory of an electronic digital computer.
(2) The result of the analysis of a sentence is to be given in the form of a structural description.

In order to minimize computing time and memory space, further constraints were devised aiming to obtain efficient recognition routines. In general these constraints were put on the structural description and much more attention has been paid to giving a rigorous definition of the structural description than to the definition of a grammar rule. Discussions of this topic can be found in Hays [2], Lecerf [3] and Plath [13]. The nature of grammatical rules is never emphasized and often the structure of the rules is not even mentioned. Nevertheless the logical priority order of the operations is the following:

—the recognition routine traces a derivation of a sentence S according to the grammar.

—the derivation of *S* provides a structural description of *S.* The structural descriptions generally have a simple form, that of a tree terminating in the linear sequence of the words of *S.* These trees (or associated parenthesized expressions) are unlabelled in [2], [3] and [13]. Yngve uses a complex tree with labelled nodes corresponding to well-defined rules of grammar. Other structures than trees could be used [10] in order to increase the amount of information displayed in the structural description. The question of how much information is necessary in the structural description of a given sentence in order to process it automatically (translation or storage and matching of information), has never been studied. Many authors look for a minimization of this information, which is quite unreasonable given the present status of the art; our guess is that the total amount of information available in any formalized system for a sentence *S* will never be sufficient for a completely mechanized processing of *S* and that these minimal structural descriptions will have to be considerably enriched. In Section 6 we will show that for many sentences one tree is not sufficient to describe relations between words.

Among the simple models which have been or are still used, we will quote:

Immediate constituent analysis models as developed in [7], [14], [15] and [16] where two substrings *B* and *C* of a sentence *S* can form a larger unit *A* of *S* only if they are contiguous *(A = BC).*

Categorical grammars developed by Bar-Hillel where categories of substrings are defined by means of the basic categories (Noun, Sentence) and of the immediate left or right environment.

Predictive Analysis models ([17], [18], [19]) where the grammars are built such that the recognition routine can use a pushdown storage, which is a convenient programming tool.

Dependency and projective grammars developed respectively by Hays [20] and Lecerf [3], which lead to the simple analysis programs their authors have described.

2
CHOMSKY'S CONTEXT-FREE LANGUAGES*

The grammars of the most general type we have described in the previous section can be viewed as arbitrary Turing Machines or equivalently [32] as combinatorial systems

* In the section we shall follow closely Chomsky [12] where a complete survey of these languages and their bibliography can be found.

(Semi-Thue Systems) where the sentences are derived from an axiom S by the means of a finite set of rewriting rules (productions): $\varphi \rightarrow \psi$.

The sentences are defined on the terminal vocabulary V_T as in Section 1. The strings φ, ψ are defined on a vocabulary $V = V_N U V_T$, where V_N is the non-terminal vocabulary which includes an initial symbol S meaning sentence.

The arrow between φ and ψ is to be interpreted as 'is rewritten'.

A context-free grammar is a finite set of rewriting rules $\varphi_i \rightarrow \psi_i$ where φ_i and ψ_i are strings on V such that:

φ_i is a single element of V_N
at least one φ_i is S
ψ_i is a finite string on V.

The language generated by a context-free grammar is called a context-free language.

Example

Grammar:
$$\begin{cases} S \rightarrow aSb \\ S \rightarrow c \end{cases}$$

This grammar can generate (recognize) all and only the sentences of the type

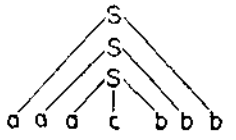
$$\underbrace{a \ldots\ldots a}_{n \text{ times}}\ c\ \underbrace{b \ldots\ldots b}_{n \text{ times}}$$

noted *a" c b",* for any *n.*

The sentence *aaa c bbb* is obtained by the following steps:

$S \Rightarrow aSb$
$aSb \Rightarrow aaSbb$ these four lines represent the derivation of the sentence.
$aaSbb \Rightarrow aaaSbbb$
$aaaSbbb \Rightarrow aaa\ c\ bbb$

The associated structural description is the following:



or equivalently $[_S a [_S a [_S a [_S c] b] b] b]$

The context-free grammars can be associated with restricted Turing Machines such as restricted infinite automata or equivalently, pushdown storage automata.

We give an informal description of the pushdown storage automaton (PDS automaton). For a more precise description see Chomsky [12].

The PDS automaton is composed of a control unit which has a finite set of possible internal configurations or states $\{S_j\}$ including an initial state S_0. The control unit is equipped with a reading head which scans the symbols a_i of a finite string written on successive squares of an input tape which is potentially infinite and can move, let us say, only from right to left.

$\{a_i | \ 1 \leq i \leq p\}$ is the set of symbols; $V_I = eU\{a_i\}$ is the input vocabulary which includes a null element *e;* the input strings are defined on V_I.

The control unit is equipped with a second head which allows it to read and write on a storage tape which can move in either direction and is also potentially infinite. It writes on successive squares of the storage tape strings on a vocabulary: $eU\{A_k|1 \leq k \leq q\}$ which can include V_I. The storage tape vocabulary is $V_0 = eU\{A_k\}U\sigma$, where e is the null element, and σ a special symbol which is never printed out.

The squares on which the strings are written are occupied simultaneously by both a_i (or A_k or σ) and e; either infinite side of the string can be thought of as filled with the blank symbol #.

A situation Σ of the PDS automaton is a triplet $\Sigma = (a_i , S_j , A_k)$; if the PDS is in the situation Σ it is also in the situations where the elements a_i or A_k or both are replaced by e. There is an initial situation (a_i , S_0 , σ) where the input head is positioned on the leftmost square of the input string, and the storage head on the symbol σ.

A computation starts in an initial situation and is directed by a finite number of instructions I, $\Sigma \rightarrow (S_r, x)$; when the automaton returns to the initial situation the first time, $x = \sigma$.

From the situation Σ, where the automaton is in state S_j, the automaton switches into the state S_r and moves its input tape one square left if the first element of Σ is an a_i , otherwise (for e) the tape is not moved.

If x is a string on $\{A_k\}$, it is printed on successive squares to the right of the square scanned on the storage tape, and the latter is moved $\lambda(x)$ (length of x) squares to the left.

If $x = \sigma$ the storage tape is moved one square right, nothing is printed and the square A_k previously scanned is replaced by the blank symbol #.

If $x = e$ the storage tape undergoes no modification. After an instruction I has been carried out, the automaton is in the new situation Σ' whose first element is the symbol of the input string now being scanned, the second element is S_r, the third element can be either the same A_k if $x = e$ or A_m if $x = A_m$, or if $x = \sigma$, the rightmost symbol of A_k written on the storage tape. If in this new situation Σ' a new instruction I' can be applied (i.e. there is an I' whose left member is Σ'), the computation goes on, otherwise it is 'blocked'.

An input string is accepted by a PDS automaton if starting in an initial situation, it computes until on its first return to S_0 it is in the situation $(\#, S_0, \#)$ the storage tape is blank, and the first blank is the one at the right of the input string (the latter has been completely scanned).

A set of strings on V_I accepted by a PDS automaton will be called a push-down language.

We can define context-free grammars and pushdown automata on the same universal alphabet V_U ; we then have the following theorem by Chomsky [12] and Schützenberger [33].

Theorem 1

The pushdown languages are exactly the context-free languages.

3
EQUIVALENCES OF LANGUAGES

(1) The equivalence of context-free languages and immediate constituent languages has been proven by Chomsky [8]. He proved that for any context-free grammar there exists a grammar whose rules are all of the form $A \rightarrow BC$ or $A \rightarrow a$ where the capital letters (members of the nonterminal vocabulary) represent structures and the a's morphemes.

Sakai's model is exactly the immediate constituent model. His grammar has rules of the

form $BC = A$, his recognition routine builds, for a given sentence, all binary trees compatible with the grammar.

(2) The equivalence of Bar-Hillel's categorical grammars and context-free grammars is proved in [22].

(3) Theorem 1 proves the equivalence of the predictive analysis and the context-free analysis. The recognition routine described in Kuno and Oettinger can be schematized by PDS automaton of Section 2 in the following way: $\{a_i\}$ is identified with the set of syntactic word classes $\{s_i\}$ given by a dictionary; $\{A_k\}$ is identified with the set of predictions $P = \{P_i\}$. The symbol S, meaning sentence and π meaning period (end of sentence) are P_i's. The set I of instructions contains:

$$
\begin{aligned}
&I_1 : (e, S_0, a) \rightarrow (S_1, e)\\
&I_2 : (e, S_1, e) \rightarrow (S_2, S)\\
&I_3 : (s_j, S_2, P_k) \rightarrow (S_{Pk}, \sigma)\\
&I_4 : (e, S_{Pk}, e) \rightarrow (S_2, x)\\
&I_F : (s_j, S_2, \sigma) \rightarrow (S_0, \sigma)
\end{aligned}
$$

The set of the states is $\{S_0, S_1, S_2, [S_{Pk} | P_k \in P]\}$.

The device computes as follows:

I_1 and I_2 are initialization instructions; I_2 places on the storage tape the prediction S (sentence).

To I_3, I_4 corresponds the use of the grammar rules; P_k is the rightmost prediction on the storage tape; s_j is the syntactic class of the scanned word on the input tape. If s_j and P_k are compatible then by an I_3 the automaton switches into the state S_{Pk} and erases P_k; then by an I_4 it switches back into the state S_2 and prints a string x of predictions which is a function of P_k and s_j (x may be null).

I_F ends the computation, it comes after an I_3 where s_j was compatible with the 'bottom' prediction π (period); π was erased, then I_F applied and leaves the automaton in the situation (#, S_0, #), where the storage tape is blank and the string accepted.

In a situation (e, S_{Pk}, e) following a situation (s_j, S_2, P_k) there may be different possible strings x corresponding to a single pair (s_j, P_k); the automaton, which is non-deterministic in this case, will choose one at random; if its choices are right all along the computation, the input string will be accepted, if not the computation will block. These conventions do not affect the class of languages accepted by the automaton. Intuitively, an acceptable string may be rejected several times because of a wrong guess, but there exists a series of right guesses that will make the automaton accept this string.

The actual device gives as an output a syntactic role r_n for each s_j, where r_n provides a structural description; this does not affect the class of languages accepted by the PDS automaton.

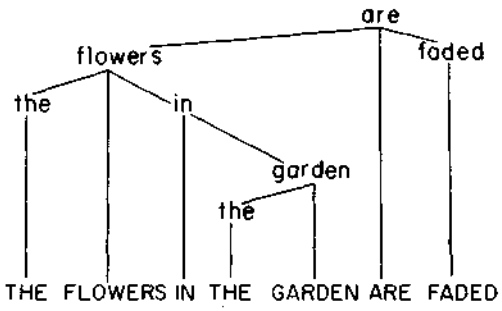
The automaton described above is precisely one which was previously used for predictive analysis, where only one (so-called 'most probable') solution (acceptance) was looked for [23]. The new scheme [17] gives all possible solutions for an input sentence. It can be considered as a monitoring program which provides inputs for the PDS automaton described above and enumerates all possible computations the automaton has to do for every input string. If a sentence contains homographs then the corresponding input strings are enumerated and fed successively into the automaton; the latter instead of making a guess when in a non-deterministic situation, tries all of them; they are kept track of by the moni-

toring program; when a computation blocks, the storage tape (subpool) is discarded and a new computation is proposed to the automaton.

4
DEPENDENCY LANGUAGES

We will turn to the dependency grammars as defined in Hays [20]. The linguistic conception originated by Tesnière differs from that of Immediate Constituent Analysis; here the morphemes are connected in terms of the intuitive notions of governor and dependent:

Example



The two basic principles which determine the shape of the dependency model are quoted as follows: ([20] pp. 3, 4).

(1) Isolation of word order rules from agreement rules.

(2) Two occurrences (words) can be connected only if every intervening occurrence depends, directly or indirectly, on one or the other of them.

(1) Corresponds to the fact that recognition routine and grammar are separated.

(2) Defines both grammar and language. The grammar consists of a set of binary relations between a governor and a dependent. Two occurrences can be connected only if a certain contiguity holds: IN and GARDEN are connected only if the (direct) dependency GARDEN-THE holds; FLOWERS and ARE can be connected only if the indirect dependency IN-THE and the direct one FLOWERS-IN, hold.

Theorem

The dependency languages are exactly the context-free languages.

This theorem has been proven by Gaifman and independently by the author. Gaifman [34] has obtained a stronger result showing that the set of the dependency trees is a proper subset of the set of the context-free trees.

We give below the part of our proof which yields the following result; the dependency languages are context-free languages.

We now construct a PDS automaton of the type of Section 2 which accepts the dependency languages. Like the previous machine, it will not give a structural description of the analysed string but the restriction that the device be normative has no effect on the class of accepted strings.

Let $V = \{a_i\}\ i \neq 0$ be the vocabulary of the language. We will take as the input vocabulary of the automaton $V_I = VUe$.

Let $V_0 = eU\{A_k\}U\sigma$, $k \neq 0$ be the output vocabulary where the A_k's are syntactic classes.

The set $\{I\}$ of instructions contains

$$
\begin{aligned}
I_1 &: (e, s_0, \sigma) \rightarrow (S_D, e)\\
I_2 &: (a_i, S_D, e) \rightarrow (S_D, A_i)\\
I_3 &: (e, S_j, A_j) \rightarrow (S_j, \sigma)\\
I_4 &: (e, S_i, Ak) \rightarrow (S_j, \sigma)\\
I_5 &: (e, S_j A_k) \rightarrow (S_k, \sigma)\\
I_6 &: (e, S_j, e) \rightarrow (S_D, A_i)\\
I_7 &: (e, S_s, \sigma) \rightarrow (S_0, r)
\end{aligned}
$$

The main operation of the computation is to connect a dependent to its governor; when this is done the dependent is erased from the storage.

I_1 initializes the computation.

I_2 is a dictionary look-up instruction; A_i a syntactic class of a_i is printed on the storage tape.

I_3 guesses that there is a connection to be made with the A_i immediately to the left of the scanned A_j, A_j is erased but remembered since the automaton switches into a corresponding state S_j.

I_4 and I_5 compare the syntactic classes A_j and A_k: the automaton switches into either the state S_j or S_k corresponding to the fact that either A_j or A_k is governor. For a given pair (A_j, A_k) there is generally either an I_4 or an I_5 according as A_j or A_k is governor (in case the agreement is unambiguous). If they cannot be connected at all, the machine stops and the string is not accepted. If the next accessible A_r on the storage tape is to be connected with A_j or A_k then an instruction: $(e, S_j, A_r) \rightarrow (S, \sigma)$ or $(e, S_k, A_r) \rightarrow (S, \sigma)$ can be applied (either of these instructions can be an I_4 or an I_5). If the next accessible A_r is not to be connected with A_j or A_k then instructions I_6 and I_2 are applied. Before the automaton uses an instruction I_5 where it forgets A_j it has to make the guess that no dependent on A_j will come next from the input tape.

I_6; the automaton transfers A_j from its internal memory to the storage tape, it switches into state S_D where it will use an instruction I_2.

I_7 is a type of final instruction; from the state S_s where the automaton remembers the topmost governor A_s, it switches into the state S_0 after the direct dependents of A_s have been connected (i.e. erased); the situation following I_7 will be (#, S_0, #).

This non-deterministic automaton is equivalent to a recognition routine that would look for one 'most probable' solution. As in the case of the predictive analysis discussed above a monitoring program that would enumerate all possible computations would provide all possible solutions.*

Remarks

The instructions I_3 and I_4 (or I_5 for a governor A_j (or A_k)) remember this governor in the form S_j (or S_k). If this governor were modified by the agreement then it could be rewritten S_m with $m \neq j$ (or $m \neq k$). This would not change the structure of the automaton, nor would it modify the class of accepted languages.

* The author has written in COMIT a recognition routine of this type. The automaton above is a schematization of a part of the program which gave, after one scan from left to right, all the solutions compatible with the grammar.

It might be convenient for the diagramming of a sentence to think of null occurrences having a syntactic class; then their position could be restricted (between consecutive A_iA_k, for example) and in this case too a modification of the automaton shows that the class of accepted languages is the same.

In the case of I_5 we have the following indeterminacy about the guess the automaton has to make; if the guess is wrong, namely the next coming A_r can be connected to A_k; two cases are possible: (1) A_r can be connected to A_j the governor of A_k, then the computation may still follow a path which will lead to an acceptance (in this case the sentence was ambiguous), (2) A_r cannot be connected to A_j. The string will not be accepted since the content of the storage tape will never become blank.

The cases of conjunction, double conjunction and subordinate conjunctions in the 'secondary structure' require the automaton to have states where it remembers two syntactic types. For example, in the string *acb* a conjunction *c* depends on governor *b* on its right, and the item a equivalent to *b* and preceding *c* is marked dependent on *c*. We will have the following instructions:

$$(C_0)\quad (a,S_D,e) \rightarrow (S_D\ A_a)$$
$$(C_1)\quad (c,S_D,e) \rightarrow (S_c,e)$$
$$(C_2)\quad (e, S_c, A_a) \rightarrow (S^a_c, \sigma)$$
$$(C_3)\quad (b,S^a_c,e) \rightarrow (S^a_c, A_b)$$
$$(C_4)\quad (e, S^a_c, A_b) \rightarrow (S_D, e)$$

(C_1) remembers the conjunction *c* read after A_a
(C_2) erases A_a from the storage but remembers it
(C_3) reads and looks up *b*
(C_4) connects A_a, *c*, A_b and forgets the dependents A_a and *c*.

Relationship grammars

Four classes of grammars, all based on the concept of dependency, are defined by Fitialov.

The author remarks that the languages described by these grammars are all phrase-structure languages in the sense of Chomsky [8]. According to Chomsky [8], [12] phrase-structure grammars include at least context-free and context-sensitive grammars. In his paper Fitialov mentions the use of contexts, and it is not clear which type of phrase structure grammars are Fitialov's grammars.

The construction of a P.D.S. automaton very similar to the one above, which is perfectly straightforward, shows that Fitialov's languages are context-free. They are probably powerful enough to describe the whole class of context-free languages, which remains to be proved.

Projective Languages

Lecerf [24] works on the principle that the dependency representation of a sentence S_1 and the immediate constituent analysis of the same sentence S_1 are both of interest since they show two different aspects of linguistics [3].

This mathematical theory deals with infinite lexicons and doubly structured strings are defined. We will impose the restriction of finiteness on the lexicon.

The recursive definition of the (G-structures is the following:

Let $M = \{m_i\}$, a lexicon where the m_i's are words:
A syntagma is represented by $[S_k]$; the m_i's are S_k's; S is the set of all possible syntagma $S = \{[S_k]\}$.
There is an operator represented by $([S_k])$ corresponding to each syntagma which can be applied to the right or the left of any syntagma in order to construct a new one.
Right operation $[S_j]$. $([S_k])$ is noted

$$([S_j] \,.\, ([S_k])] \in S.$$

Left operation $([S_k])$. $[S_j]$ is noted

$$[([S_k]) \,.\, [S_j]] \in S.$$

Example

[([([the]) . [man]]) . [[hit] . ([([the]) . [ball]])]]

Lecerf proved that the operation of erasing the parentheses provides the tree of the immediate constituent analysis where the dots are the nonterminal nodes. On the other hand, the operation which consists of erasing the brackets and dots provides the tree of the dependency analysis.

We will point out some properties of this model. The language defined by right or left adjunction of an operator to a syntagma is obtained when we erase the structure markers (parentheses, brackets, dots). These adjunctions then reduce to the simple operation of concatenation. The language defined is the set of all possible combinations of words. It is the monoid *C(M)*.

Clearly what is missing is a set of rules which would tell which syntagma and which operators can be combined, but this problem is not raised.

If, following the author, we admit that his model characterizes both the dependency languages and the immediate constituent languages, we have two good reasons to think of the 'G-structures' as being context-free, but no evidence at all. In this case the 'G-structures' would be redundant since Gaifman gave an algorithm which converts every dependency grammar into a particular context-free grammar.

Yngve's model

The grammar described in Yngve *[6]* consists of the following types of rules defined as a vocabulary $V = V_N U V_T$

(1) $A \rightarrow a$

(2) $A \rightarrow BC$

(3) $A \rightarrow$ B ... C where any single element of V can occur between B and C in rule (3).

The restriction of the depth hypothesis makes the language a finite state language [6]. Without depth, rules of type (1) and (2) show that the language is at least context-free.

Matthews [25] studied a special class of grammars; grammars containing rules of types (1), (2), (3) are shown to be a subclass of Matthews' 'one-way discontinuous grammars', which in turn are shown to generate context-free languages.

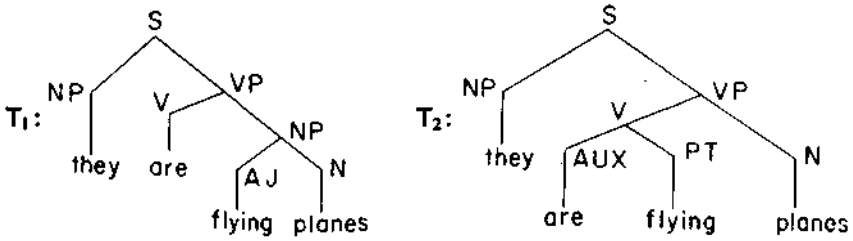
The structural description provided by these rules are not simple trees and cannot be compared to the other structures so far described.

5
APPLICATIONS OF THE GENERAL THEORY OF C.F. GRAMMAR

The context-free grammars, have found applications in the field of programming languages. We extend here remarks already made by Chomsky [9] and Ginsburg and Rose [26] to the case of natural languages. The two theorems mentioned below derive from the results obtained by Bar-Hillel *et al.* [27].

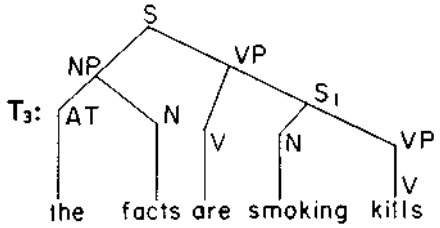
Ambiguity

An empirical requirement for a grammar is that of giving n structural descriptions for an *n*-ways ambiguous sentence. A grammar of English will have to give, for example, two analyses of the sort given below, for the sentence: (A) They are flying planes



When constructing a *c-f* grammar for natural languages the number of rules soon becomes very large and one is no longer able to master the interrelations of the rules: rules corresponding to the analysis of new types of sentences are added without seeing exactly what are the repercussions on the analysis of the former ones. We use here an example mentioned in [17]. The grammar used at the Harvard Computation Laboratory contains the grammatical data necessary to analyse the sentence (A) in two different manners. Independently it contains the data necessary to analyse the sentence (B) in the manner shown below:

(B) The facts are smoking kills



The dictionary shows that TO PLANE is also a verb and KILL also a noun.

The result of the analysis of sentences (A) and (B) is three solutions for each of them (the three trees described above: T_1 , T_2, T_3).

Any native speaker of English will say that (A) is twice ambiguous and (B) is not ambiguous at all; the wrong remaining analyses are obtained because of the grammar in which rules have to be made more precise.

Obviously T_3 has to be suppressed for (A) and T_1 and T_2 for (B). This situation arises very frequently and one may raise the more general question or systematically detecting

ambiguities in order to suppress the undesirable ones. A result of the general theory of c.f. grammars is the following.

Theorem. The general problem* of determining whether or not a c.f. grammar is ambiguous is recursively unsolvable, [21].

The meaning of the result is the following: there is no general procedure which, given a c.f. grammar, would tell, after a systematic inspection of the rules, whether the grammar is ambiguous or not. Therefore the stronger question of asking what rules produce ambiguous sentences is recursively unsolvable as well. We can expect that the problem of checking an actual grammar by other means than analysing samples and verifying the structural descriptions one by one is extremely difficult.

Translation

A scheme widely adopted in the field of M.T. [5] consists of having two independent grammars G_1 , G_2 for two languages L_1 , L_2 and a transfer grammar T going from L_1 to L_2 (or from L_2 to L_1). A result of the theory of c.f. grammars is the following:

Theorem. Given two c.f. languages L_1 and L_2, the problem of deciding whether or not there exists a mapping T such that $T(L_1) = L_2$ is recursively unsolvable [26].

Of course a translation from L_1 to L_2 need not be an exact mapping between L_1 and L_2, but there may be a large sublanguage of L_2 which is not in the range of any particular grammar T constructed empirically from L_1 and L_2; conversely some sublanguage of L_1 may not be translatable into L_2 ; in any case, one should expect that the problem of constructing a practically adequate T for L_1 and L_2 is extremely difficult.

These results derived from the general theory show the interest of this theory. The results so far obtained mostly concern the languages themselves; very little is known about the structural descriptions. Studies in this field could provide decisions when a choice comes between different formal systems: for example, this theory could decide which one of the systems described above is more economical according to the number of syntactic classes, or to the number of operations necessary to analyse a sentence. Such questions, if they may be answered, require further and difficult theoretical studies on these systems.

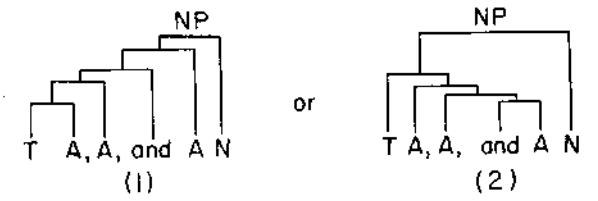
6
ADEQUACY OF CONTEXT-FREE MODELS

Many natural languages present to a certain extent, the features of context-free languages. An example of strings which are not context-free has been given by Bar-Hillel and Solomonoff:
N_1, N_2. ... AND N_k ARE RESPECTIVELY A_1, A_2...A_k where a relation holds between each noun N_i and the corresponding adjective A_i. These strings cannot be generated for any k by a context-free grammar.

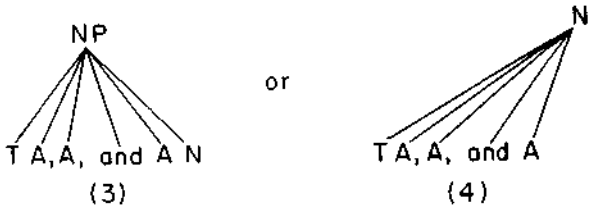
More obvious is the inadequacy of the context-free structural descriptions. Chomsky [7, 11, 12] pointed out that no c.f. grammar can generate the correct structure for a sequence of adjectives modifying a noun.

* Except in a very simple case of no linguistic interest.

c.f. structures:



correct structures:



In the cases (1) and (2) above the Noun Phrase has been given too much structure by a c.f. grammar. Very frequent too are the cases where no structure can be given at all by a c.f. grammar: the ambiguous phrase:

THE FEAR OF THE ENEMY

cannot be given two different structures showing the two possible interpretations, the same remark applies to phrases of the type:

VISITING RELATIVES.

Another case of lack of information in the structural description given by a context-free grammar is the following:

(i) THE EXPERIMENT IS BOUND TO FAIL

BOUND and TO FAIL have to be connected but nothing can tell that EXPERIMENT is subject of TO FAIL.

Once a context-free structure is given to (i) it is not very simple to give a different one to the sentence:

(ii) THE EXPERIMENT IS IMPOSSIBLE TO REALIZE

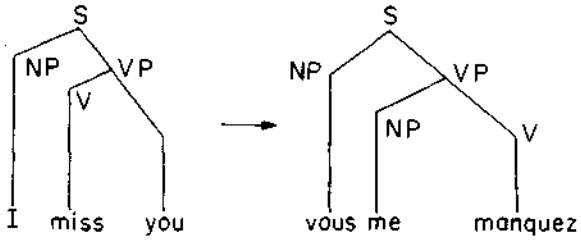
where EXPERIMENT is object of TO REALIZE.

In any case, if two different structures are given to (i) and (ii), nothing will tell any more that these two sentences are very similar.

All these examples are beyond the power of context-free grammars. Adequate treatments are possible when using transformational grammars (Chomsky) but these grammars, more difficult to construct and to use, have been disregarded in the field of M.T. on the grounds that they could not be used in a computer, which is false. The programming language COMIT uses precisely the formalism of transformations. Matthews is working on a recognition routine which is using a generative transformational grammar [28], [29].

The interest of a word for word translation being very limited [30] a scheme of translation sentence for sentence motivated the construction of grammars.

Every sentence requires to be given a structural description, and a transfer grammar maps input trees into output trees. Considered from a formal point of view, a transfer grammar is precisely a transformational grammar.



The construction of a transfer grammar between two context-free grammars raises serious problems.

Let us consider the following example of translation from English to French taken from Klima [31].

(a) HE DWELLED ON ITS ADVANTAGES

an almost word for word translation gives the French equivalent:

(a') IL A INSISTE SUR SES AVANTAGES

but let us consider the passive form (p) of the sentence (a):

(p) ITS ADVANTAGES WERE DWELLED ON BY HIM

in French (a') has no passive form and (b) has for translation the sentence (a'). What is required for the translation of (p) is either a transformation of a French passive non-sentence (p') (obtained almost word for word):

(p') *SES AVANTAGES ONT ETE INSISTE SUR PAR LUI

into the sentence (a') or a transformation of (p) into (a), made before the translation. The two solutions are equivalent from the point of view of the operations to be carried out but the second seems more natural.

In the latter case, since the passive sentences are described by the means of the active sentences and a transformation, no context-free description of the passive sentences is longer required in the source grammar. Many other cases of the type above show that the use of a transformational source grammar will simplify the transfer grammar, moreover Chomsky has shown that transformational grammars simplify considerably the description of languages. These are two good reasons for M.T. searchers to become interested in models which are less limited than context-free models.

ACKNOWLEDGEMENTS

I am indebted to Noam Chomsky for important remarks, to Barbara Hall for her remarks and her editorial assistance, to Hugh Matthews and Victor Yngve for several suggestions.

The preparation of this work was supported in part by the Centre d'Etudes pour la Traduction Automatique (Section de PARIS) and by a UNESCO fellowship granted for a stay at Harvard University and the Massachusetts Institute of Technology (Research Laboratory of Electronics).

This work is also supported in part by the National Science Foundation, and in part by the U.S. Army Signal Corps, the Air Force Office of Scientific Research, and the Office of Naval Research.

REFERENCES

[1] Y. BAR-HILLEL: *Language* (1953), 29, 47-48.
[2] D. G. HAYS: *Grouping and Dependency Theory,* The Rand Corporation (1960).
[3] Y. LECERF: Programme des Conflits, Modèle des Conflits, Euratom, Rapport Grisa No. 4 (1960).
[4] A. SESTIER, and L. DUPUIS: La Place de la Syntaxe dans la Traduction Automatique des Langues, Ingenieurs et Techniciens, Paris (1962).
15] V. H. YNGVE: *Mechanical Translation,* 1959, 4, No. 3, 59.
[6] V. H. YNGYE: First International Conference on Machine Translation, Teddington, England (1961).
[7] N. CHOMSKY: *Syntactic Structures.* Moutons', Gravenhage (1957).
[8] N. CHOMSKY: *Information and Control* (1959), 2, 137-167.
[9] N. CHOMSKY: On the Notion 'Rule of Grammar'. Symposia in Applied Mathematics, Vol. XII, American Mathematical Society, Providence, R.I. (1961).
[10] L. TESNIERES: *Syntaxe Structurale.* Klincksieck, Paris (1960).
[11] N. CHOMSKY: *IRE Trans. Infor. Theory, IT2,* 113-114 (1956).
[12] N. CHOMSKY: *Handbook of Mathematical Psychology.* (Ed. by D. Luce, E. Bush, E. Galanter) (1962).
[13] W. PLATH: First International Conference on Machine Translation, Teddington, England (1961).
[14] Z. S. HARRIS: *Methods in Structural Linguistics.* University of Chicago Press (1951).
[15] Z. S. HARRIS: *Language* (1957), 33, 283-340.
[16] R. WELLS: *Language,* 1947, 23, 81-117.
[17] S. KUNO and A. OETTTNGER: Multiple syntactic analyser—Harvard Computation Laboratory (to appear in the proceedings of the I.F.I.P. conference, Munich, 1962).
[18] A. OETTINGER: *Automatic Syntactic Analysis and the Pushdown Store.* Proceedings of Symposia in Applied Mathematics, Vol. XII, American Mathematical Society, Providence, R.I. (1961).
[19] I. RHODES: A new approach to the mechanical translation of Russian. National Bureau of Standards, Report, No. 6295 (1959).
[20] D. G. HAYS: *Basic Principles and Technical Variations in Sentence Structure Determination,* The Rand Corporation (1960).
[21] M. P. SCHÜTZENBERGER: On a family of formal power series (to be published).
[22] Y. BAR-HILLEL, C. GAIFMAN and E. SHAMIR : *Bull. Res. Council Israel* (1960), 9, F, No. 1.
[23] M. SHERRY: First International Conference on Machine Translation, Teddington, England (1961).
[24] Y. LECERF : Une représentation Algébrique de la Structure des Phrases dans diverses Langues Naturelles. Notes aux comptes-rendus de 1'Academie des Sciences Paris, 1961, 252, No. 2, 232.
[25] G. H. MATTHEWS: One-way discontinuous grammars, (to be published).
[26] S. GINSBURG and F. R. ROSE: Some recursively unsolvable problems in algol-like languages. System Development Corporation—S.P. 690 (1962).
[27] Y. BAR-HILLEL, M. PERLES and E. SHAMIR: Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung—Band 14, Heft 2 (1961).
[28] G. H. MATTHEWS and S. ROGOVIN: *German Sentence Recognition, Mechanical Translation,* Vol. 5, No. 3, M.I.T. Cambridge, Mass. (1958).
[29] G. H. MATTHEWS: 1961 First International Conference on Mechanical Translation, Teddington, England (1961).
[30] E. S. KLIMA: First International Conference on Machine Translation, Teddington, England (1961).
[31] E. S. KLIMA: Correspondence at the grammatical level. Q.P.R. No. 64, R.L.E., M.I.T. Cambridge, Mass. (1962).
[32] M. DAVIS: *Computability and Unsolvability.* McGraw Hill, New York (1958).
[33] M. P. SCHÜTZENBERGER: *Some Remarks on Chomsky's Context-free Languages.* Q.P.R. No. 63, R.L.E., M.I.T., Cambridge, Mass. (1961).
[34] H. GAIFMAN: *Dependency Systems and Phrase Structure Systems.* Rand Corporation—P. 2315(1961).
[35] YA. FITIALOV: *On Models of Syntax for Machine Translation.* J.P.R.S. 13197—28 March 1962.