INFORMATION AND CONTROL 1, 91–112 (1958)

Finite State Languages*

NOAM CHOMSKY

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

GEORGE A. MILLER

Harvard University, Cambridge, Massachusetts

A finite state language is a finite or infinite set of strings (sentences) of symbols (words) generated by a finite set of rules (the grammar), where each rule specifies the state of the system in which it can be applied, the symbol which is generated, and the state of the system after the rule is applied. A number of equivalent descriptions of finite state languages are explored. A simple structural characterization theorem for finite state languages is established, based on the cyclical structure of the grammar. It is shown that the complement of any finite state language formed on a given vocabulary of symbols is also a finite state language, and that the union of any two finite state languages formed on a given vocabulary is a finite state language; i.e., the set of all finite state languages that can be formed on a given vocabulary is a Boolean algebra. Procedures for calculating the number of grammatical strings of any given length are also described.

In the vast majority of communication situations the messages that are exchanged consist of strings of symbols. It is possible to imagine a code that uses only one symbol per message, but few situations are so

* This work was supported in part by the Army (Signal Corps), the Navy (Office of Naval Research), the Air Force (Office of Scientific Research and Operational Applications Laboratory, Air Research and Development Command), the National Science Foundation, and the Social Science Research Council (Committee on Mathematical Training of Social Scientists), and appears as report number AFCRC-TR-58-50, ASTIA Document Number AD 146781.

rigidly structured that it is possible to know in advance everything that can happen and to provide a special symbol to represent it, and still fewer situations are so impoverished for outcomes that the number of distinct symbols required would be small enough for accurate production and recognition of each symbol. Strings of symbols may take more time to transmit, but they have many advantages: the strings can be composed from a relatively small set of easily discriminable symbols, novel strings can be composed to describe novel situations as they arise, and the variety of strings available is quite large, since the number grows exponentially as the length of the string increases.

In natural languages, however, not all the possible strings of symbols are actually used. This fact is usually referred to as the "redundancy" of natural languages. Some strings do not occur because we have no occasion to use them. But many strings are explicitly prohibited—they are not admissible utterances in the language. Thus we are faced with the problem of specifying in some simple way which strings are admissible, or well-formed, or grammatical and which are prohibited, meaningless, or ungrammatical. For natural languages, this specification is usually assumed to be the task of the grammarian. He must find a grammar for the admissible strings. It is not obvious that this problem must always have a solution.

A grammar is a set of rules—preferably a finite set, if we expect finite automata to learn them—that specify the grammatical strings of symbols. Now there are a great many different ways to state a set of rules. The rules as stated in the traditional grammar books do not lend themselves to logical analysis, and so it is natural to search for some alternative method of description that will be more compatible with our modern methods of describing communication processes in general. For example, one possible method for describing a grammar is in terms of a program for a universal Turing machine. In this paper, however, we shall limit the discussion to a less powerful type of device which does not have an infinite memory and which must generate the strings in a fixed order, from past to future (or left to right). The memory of the generator is limited entirely to the state that it finds itself in at any given moment and we assume that there is only a finite number of these states available. Each new symbol follows (or is placed to the right of) the string of symbols already generated. Such a generator is here called a finite state grammar and the set of strings that it can produce is a finite state language. The purpose of this paper is to examine the properties of such a language.

In recent years the representation of a message source by a stochastic process, usually a finite Markov process, has become a familiar procedure (Shannon, 1948). A finite Markov process is essentially a finite state generator that is supplemented by probability distributions for the choices available in each state. There are two important questions we could ask about such generators: (1) What properties characterize languages produced by such generators, and (2) Do natural languages have these properties? It seems clear that a finite state grammar is not adequate for most natural languages (Chomsky, 1956), so the answer to the second question is negative. Nevertheless, the mathematical properties of such processes are well suited to the needs of communication engineers and it is likely that they will continue to be of interest in many applications of communication theory. Thus the first question is not without interest; it provides the focus of the present discussion.

One property of these finite state models is that they generate indefinitely long strings of symbols. The mathematical convenience of passing to the limit is clear, but the practical fact remains that linguistic messages are usually rather brief and are broken into units that we recognize, more or less vaguely, as words and sentences. Some attention has been paid to the segmental nature of linguistic messages and to the problems of scansion when messages must be encoded and decoded without delay (Schützenberger, 1956). It would seem that only a minor modification of the finite state model is required in order to permit it to generate units similar to words or sentences. Mandelbrot (1954) pointed out that we can select some recurrent symbol or state of the system and identify the occurrence of this symbol or state with the end of one segment and the beginning of the next. If we refer to these subsegments as sentences, then a message is a string of one or more sentences.

In the present paper we examine some of the consequences of introducing segmentation in this way and explore some of the properties of the sets of sentences that can be produced. We begin by showing the structural equivalence of several ways of describing the model. We then show that the languages (i.e., sets of sentences) produced by such models form a Boolean algebra, present a simple way of characterizing these languages, and conclude with a procedure for computing the number of different sentences as a function of their length.

BASIC CONCEPTS

If we are given a finite vocabulary of m words, it is possible to arrange them into an infinite variety of different strings of words. There will be m different strings one word long, m^2 different strings two words long, etc., and in general there will be m^λ different strings of exactly length λ and $m(m^{\lambda+1} - 1)/(m - 1)$ strings of at most length λ. Let the set of all strings in the given vocabulary be represented by "U." We will define a *language* as any finite or infinite subset of U. If a string occurs as a member of some particular language L, we will call it a "sentence" in that language. We will define a *grammar* as a finite set of rules for selecting a particular language from U. (Although U is only denumerably infinite, the set of all subsets of U is nondenumerably infinite. Thus the set of possible languages in a given vocabulary is nondenumerable. A grammar, however, is of finite length, so the set of all grammars is denumerable. Therefore, not all languages can have grammars that generate them.)

We will say that two languages are *structurally equivalent* if both contain exactly the same set of sentences. And we shall say that two grammars are structurally equivalent if they generate structurally equivalent languages. By suggesting that the equivalence is one of structure, we wish to leave open the question as to whether languages are completely equivalent if and only if the same probability is assigned to every sentence in both. In what follows we shall ignore probabilities and deal only with the full set of possibilities in any language. Thus, no confusion will result if we speak hereafter of equivalence when we should use the clumsier phrase, "structural equivalence."

Every language can be represented in the form of a *tree*. At the root of the tree is a point from which all sentences start. Each word that can occur as the initial word in some sentence is represented by a branch leaving this initial point. At the end of the branch representing any particular word will be another set of branches representing each of the words that can follow the first. This process of arborization continues until every possible sentence is represented by some path through the tree; or, if the language contains an infinite set of sentences, the tree will continue indefinitely. If we insist that no word is ever represented by two branches leaving the same point, then every sentence will specify a unique path through the tree. If we identify the branch points as "states" of the system, then we can think of the tree as an infinite state generator of the language. Every language can be produced by such an infinite state generator. Only if the process is finite, however, can we refer to the tree as a grammar.

A *finite state grammar* G is determined by a set of internal states $S_0, \cdots, S_n$ and transition symbols $W_0, \cdots, W_m$. A binary associative operation of *concatenation* (symbolized $\wedge$) is defined on these transition symbols; $W_{a_1} \wedge \cdots \wedge W_{a_n}$ is the *string* formed by concatenating the symbols $W_{a_1}, \cdots, W_{a_n}$ from left to right in this order. For generality, we assume that each grammar has an identity element W_0 with the property that for every string X, $X \wedge W_0 = W_0 \wedge X = X$. We will see that this element is often, but not always, eliminable without changing the character of the grammar. S_0 is called the *initial state*. The set $\{W_1, \cdots, W_m\}$ is called the *vocabulary* of G. Each ordered pair (j, k), $0 \leqq j \leqq m$, $0 \leqq k \leqq n$, defines a *grammatical rule*. If the grammatical rule (j, k) is associated with state S_i, this indicates that when the grammar is in state S_i the word W_j can be produced, switching the grammar into state S_k. For each i, the state S_i can be represented as a set of triples $\{(i, j, k)\}$, where (j, k) is a grammatical rule. When the grammar G begins in the initial state and moves through a sequence of states, returning to S_0 for the first time, it produces a sentence consisting of a string of words in the order in which they were selected with the successive transitions. Thus a string of words $W_{a_1} \wedge \cdots \wedge W_{a_q}$ is a sentence generated by G just in case there is a sequence of words $(W_{b_1}, \cdots, W_{b_r})$ and a sequence of states $(S_{c_1}, \cdots, S_{c_{r+1}})$ of G such that (i) $c_1 = c_{r+1} = 0$; (ii) $c_i \neq 0$ for $1 < i < r + 1$; (iii) for each i such that $1 \leqq i \leqq r$, (c_i, b_i, c_{i+1}) is one of the triples corresponding to S_{c_i};

$$\text{(iv)} \qquad W_{a_1} \wedge \cdots \wedge W_{a_q} = W_{b_1} \wedge \cdots \wedge W_{b_r}.$$

The language L_G generated by G is the set of such sentences. A *finite state language* is any language generated by a finite state grammar. In particular, the universal language U and the null language are finite state languages, the latter generated by any grammar containing no path which both initiates and terminates with S_0.

It is often convenient to represent such grammars by "state diagrams." Each state is represented by a point and each rule by an arrow running between points. If the triple (i, j, k) corresponds to S_i, then the point corresponding to S_i will be connected to the point corresponding to S_k by an arrow labeled W_j.

It is clear that the representation of a state of a finite state grammar as a set of triples is somewhat redundant, since what is important about each state is just the set of grammatical rules associated with it. Suppose that two states S_i and S_j of the grammar G are associated with the same set of grammatical rules. We can then construct the grammar G' containing the state S_q not contained in G and differing from G only in that for some word W_k of G, (i, k, q) is added to S_i, and (q, k, q) to S_q. Obviously, G' and G generate the same language,[1] and in G', the states corresponding to S_i and S_j are associated with different sets of grammatical rules. It is obvious, then, that the set of finite state languages will not be reduced if we consider only grammars in which no two states are associated with the same set of grammatical rules. In other words, we can henceforth consider a state to be simply a set of pairs $\{(j, k)\}$, where (j, k) is a grammatical rule; i.e., if the state S_i contains (j, k), then there is a transition from S_i to S_k with the symbol W_j emitted and every state is uniquely identified by the set of grammatical rules associated with it.

SOME STRUCTURAL EQUIVALENCES

Given any set F_i of finite state grammars, we will denote by $L(F_i)$ the set of languages generated by grammars of F_i. We will now consider the effects of restricting the form of grammars in various ways on the set of languages that can be generated. Let F_1 be the set of unrestricted finite state grammars described above. Let F_2 be the set of finite state grammars with the additional restriction that if $(i, 0) \in S_j$, then $i = 0$. In other words, only the identity element can end a sentence.

THEOREM 1. $L(F_1) = L(F_2)$.

To prove this it is only necessary to show that for any grammar G of F_1 with states $S_0, \cdots, S_n$ it is possible to construct a grammar G' of F_2 equivalent to G. To construct G' simply add to G a state S_{n+1} containing only $(0, 0)$, and in each state of G replace each rule $(k, 0)$ by $(k, n + 1)$. It is evident that X will be a sentence of L_G if and only if $X \wedge W_0 = X$ is a sentence of $L_{G'}$.

In grammars of F_2, the identity element plays the role of a period ending a sentence. This punctuation mark in the representation of a sentence is convenient. Since the end of a sentence is defined only by recurrence of state S_0 there is a real possibility that we will not know from the sequence of transition symbols whether or not the sentence has

[1] Since S_q operates as an absorbing barrier, it is irrelevant for the generation of sentences.

ended. This is essentially the problem of scansion discussed by Schützenberger.

The analogy with terminal punctuation is made complete in grammars of type F_3 in which all sentences end with W_0, which occurs in no other position. That is to say, in grammars of F_3 if $(i, j) \in S_k$, then $i = 0$ if and only if $j = 0$.

THEOREM 2. $L(F_2) = L(F_3)$.

Again it is necessary to prove only the inclusion from right to left. Suppose that we have the grammar G of F_2 with states $S_0, \cdots, S_n$. Perform the following preliminary construction. First, if for some k, $(0, k) \in S_k$, delete this rule. If now for some r and k, $(r, k) \in S_k$, then form the new state S_{k_1} containing the grammatical rules of S_k and (r, q), where S_q is a new state containing only (r, q). Then replace (r, k) by (r, k_1) in states S_k and S_{k_1}. Let $\bar{G}$ be the grammar formed by applying this construction throughout. Evidently, $L_G = L_{\bar{G}}$. Suppose now that the states of $\bar{G}$ are renumbered as $S_0, \cdots, S_m$, and the rules are correspondingly revised. We construct the grammar G' with states $T_0, \cdots, T_m$ defined as follows: (1) if $(i, k) \in S_j$ and $i \neq 0$, then $(i, k) \in T_j$ (that is, in G' there is transition from T_j to T_k with the symbol W_i); (2) if $(0, k) \in S_j$ and $k \neq 0$, then $T_j \supset T_k$; and (3) if $(0, 0) \in S_j$, then $(0, 0) \in T_j$. Definition (1) insures that G' will include all rules of G which do not contain W_0 and (3) carries terminal punctuation from G to G'. Definition (2) assigns to T_j all rules of T_k in case it is possible to move from S_j to S_k by the vacuous transition symbol W_0. Clearly $L_G = L_{G'}$.

It is a consequence of Thm. 2 that any language formed by deleting a particular word wherever it occurs in the sentences of a language $L(F_3)$ is again a language of $L(F_3)$. It is only necessary to replace the transition symbol for this word by the identity element in the grammar that generates the original language, thus giving an F_2 grammar. It follows from the theorem that there is an equivalent F_3 grammar that will generate just those sentences with the particular word deleted.

In an F_3 grammar, W_0 occurs only as the transition into S_0, so we can tell when a sentence ends if we know the sequence of transition symbols produced by the grammar that "spell" the sentence. It may still be possible, however, to produce the same sentence by two different state sequences. Consequently, we may not be able to determine the present state of the grammar by observing the sequence of symbols it has so far produced. We can remove this ambiguity by imposing the restriction that a transition symbol can appear in no more than one of the grammatical rules belonging to a given state. This additional restriction defines the set of grammars F_4. F_4 is the set of grammars with terminal "punctuation" but no other occurrences of the identity element, and with the restriction that if $(i, h) \in S_j$ and $(i, k) \in S_j$, then $h = k$. The sequence of states is now uniquely determined by the sequence of produced transition symbols.

THEOREM 3. $L(F_3) = L(F_4)$.

Again, it is only necessary to show that for every grammar G of F_3 there is an equivalent unambiguous F_4 grammar G'. That this is true can be seen by the following reasoning. Suppose that we are given $G \in F_3$, which produces L_G. We can represent L_G in the form of a tree, which we know to have the property that every string of symbols is associated with a unique path through the tree. We then associate the branch points in the tree with the states in G; if more than one state in G can be associated with a given branch point in the tree, we designate that branch point as a compound state containing all the rules contained in all the states of G that could be associated with it. Whenever two branch points in the tree are associated with the same state or the same set of states in G, we know that the parts of the tree which follow must be identical. If, proceeding from the root of the tree along any path, we come to a second occurrence of the same state or compound state, we know the path has become periodic and that we can identify the first occurrence with the second, thus creating a "loop" and terminating that branch of the tree. Since the number of compound states is finite, every path must have some recurrent state, so the entire tree will be converted into a finite diagram. The procedure of terminating the branches in loops does not introduce any ambiguities, so the resulting diagram will be unambiguous. It is then a simple task to write the rules of G' from this diagram, where the states of G' are simply the compound states created in the process of forming the tree. In this way we can always construct an unambiguous grammar $G' \in F_4$.

We now proceed to give a more careful proof of Thm. 3, along the lines sketched above. Suppose that $S_0, \cdots, S_n$ are the states of a finite state grammar G of F_3 with vocabulary $W_1, \cdots, W_m$. We now construct $T_1, \cdots, T_{2^n}$, each of which is correlated arbitrarily but uniquely to one of the nonempty subsets of the set $\{S_1, \cdots, S_n\}$, except for T_1

which is correlated to S_0. For each i, let K_i be the set-theoretic union of the states correlated to T_i. Thus K_i is a set of grammatical rules, i.e., pairs (j, k) where j is the index of a transition symbol and k the index of a state. For each transition symbol W_j that appears in K_i define $f(j, i)$ as the set of states S_k such that $(j, k) \in K_i$. Now define T_i as the set of pairs (j, r), where T_r is correlated to $f(j, i)$.[2]

Construct the grammar G' with the vocabulary $W_1, \cdots, W_m$, indexed in the same order as in G, and with states selected from among the sets $T_1, \cdots, T_{2^n}$ as follows. T_1 is assigned to G' as its initial state. Suppose that T_i has been assigned to G' and that $(i, r) \in T_i$. Then T_r is assigned to G'. No other T_j's are assigned to G'. Thus the T_j's assigned to G' play the role of the branch points in the tree in the informal discussion above.

Obviously G' is a finite state grammar. Furthermore it is unambiguous. Suppose in fact that $(j, k) \in T_i$ and $(j, h) \in T_i$. Thus T_k and T_h are correlated to the set of states $f(j, i)$ of G. But this correlation was one-one. Hence $k = h$. Consequently G' is an F_4 grammar. We now show that $L_G = L_{G'}$.

Suppose that $W_{a_1} \wedge \cdots \wedge W_{a_k}$ is a sentence of L_G generated by the sequence of states $(S_{t_1}, \cdots, S_{t_{k+1}})$ of G. Thus $t_1 = t_{k+1} = 0$; $t_i \neq 0$ for $1 < i < k + 1$; and $(a_i, t_{i+1}) \in S_{t_i}$, for each i.

T_1 has been assigned to G' and correlated to S_0. By assumption, $(a_1, t_2) \in S_0 = K_1$. Hence $f(a_1, 1)$ is defined and correlated to some T_{r_2}. Consequently, $(a_1, r_2) \in T_1$, and T_{r_2} has been assigned to G'. Thus the sequence of states (T_1, T_{r_2}) of G' generates W_{a_1}.

By assumption, $(a_2, t_3) \in S_{t_2}$. But $S_{t_2} \in f(a_1, 1)$, which is correlated to T_{r_2}. The set-theoretic union of the sets of $f(a_1, 1)$ has been defined as K_{r_2}; consequently, $(a_2, t_3) \in K_{r_2}$. $f(a_2, r_2)$ is thus defined and correlated to some T_{r_3}. Consequently $(a_2, r_3) \in T_{r_2}$, and T_{r_3} has been assigned to G', and the sequence of states (T_1, T_{r_2}, T_{r_3}) generates the string $W_{a_1} \wedge W_{a_2}$.

Similarly, we construct a sequence $(T_1, T_{r_2}, \cdots, T_{r_k})$ of states of G' which generates $W_{a_1} \wedge \cdots \wedge W_{a_{k-1}}$. By assumption $t_{k+1} = 0$. As above,

[2] Notice that every set $f(j, i)$ which is defined in this way has a T_r correlated to it. This follows from the fact that in G, which is by assumption an F_3 grammar, W_0 occurs as the only transition to S_0, and nowhere else. Hence $f(0, i)$ contains only S_0, and for $j \neq 0, f(j, i)$ does not contain S_0. Some T_r has been correlated to S_0 and to each set of S_i's $(1 \leqq i \leqq n)$; we do not distinguish here between a state and the unit class of that state.

we show that $(a_k, S_0) \in K_{r_k}$ and hence $(a_k, T_1) \in T_{r_k}$, since T_1 is correlated to S_0. Consequently, the sequence $(T_{r_1}, \cdots, T_{r_{k+1}})$ of states of G' generates $W_{a_1} \wedge \cdots \wedge W_{a_k}$, where $r_1 = r_{k+1} = 1$, and $r_i \neq 1$ for $2 \leqq i \leqq k$.[3] Thus $L_G \subset L_{G'}$.

Suppose now that $W_{a_1} \wedge \cdots \wedge W_{a_k}$ is a sentence of $L_{G'}$, generated by the sequence of states $(T_{r_1}, \cdots, T_{r_{k+1}})$. Thus $(a_k, r_{k+1}) \in T_{r_k}$, where $r_{k+1} = 1$, and $(a_k, 0) \in K_{r_k}$, which is the union of the sets correlated to T_{r_k}. Hence there must be some S_{t_k} among the states correlated to T_{r_k} such that $(a_k, 0) \in S_{t_k}$. The sequence of states (S_{t_k}, S_0) of G thus generates W_k. Note that the choice of S_{t_k} is not necessarily unique. Suppose, however, that $t_k = 0$. Then $S_{t_k} = S_0$ is the only state correlated to T_{r_k}, so that $r_k = 1$, $k = 1$, and $W_{a_1} \wedge \cdots \wedge W_{a_k} = W_{a_k}$ is generated by (S_{t_k}, S_0), and consequently is a sentence of L_G, as was to be proved.

Suppose that $t_k \neq 0$. By assumption, $(a_{k-1}, r_k) \in T_{r_{k-1}}$. Consequently, T_{r_k} corresponds to a set $f(a_{k-1}, r_{k-1})$ containing just those states S_j such that $(a_{k-1}, j) \in K_{r_{k-1}}$. Since S_{t_k} is one of these states, $(a_{k-1}, t_k) \in K_{r_{k-1}}$. But $K_{r_{k-1}}$ is just the union of the states correlated to $T_{r_{k-1}}$. Hence there must be an $S_{t_{k-1}}$ in the set correlated to $T_{r_{k-1}}$ such that $(a_{k-1}, t_k) \in S_{t_{k-1}}$. Again, if $t_{k-1} = 0$, the proof is completed.

Continuing in this way, we construct a (not necessarily unique) sequence of states $(S_{t_2}, \cdots, S_{t_k}, S_0)$ which generates $W_{a_2} \wedge \cdots \wedge W_{a_k}$, such that S_{t_2} is one of the states correlated to T_{r_2}, and all of $t_2, \cdots, t_k$ are distinct from 0. By assumption $(a_1, r_2) \in T_{r_1} = T_1$. Hence T_{r_2} is correlated with a set $f(a_1, 1)$ containing just those states S_j such that $(a_1, j) \in K_1$. But $K_1 = S_0$ and $S_{t_2} \in f(a_1, 1)$. Consequently, $(a_1, t_2) \in S_0$. We can thus form a sequence of states $(S_0, S_{t_2}, \cdots, S_{t_k}, S_0)$ $(t_i \neq 0)$ which generates $W_{a_1} \wedge \cdots \wedge W_{a_k}$ and is thus a sentence of L_G. Consequently $L_G = L_{G'}$. A corollary follows immediately from Theorem 3.

Corollary. *If $L_1, L_2 \in L(F_4)$, then $L_1 \cup L_2 \in L(F_4)$.* In fact, let G_1 be an F_4 grammar of L_1 with states $S_0, \cdots, S_n$ and vocabulary $W_1, \cdots, W_m$, and let G_2 be an F_4 grammar of L_2 with states $T_0, \cdots, T_{n'}$ and vocabulary $V_1, \cdots, V_{m'}$. Form the new grammar $\bar{G}$ with states $R_0, \cdots, R_{n+n'}$ and vocabulary $Y_1, \cdots, Y_{m+m'}$, where (i) $R_1, \cdots, R_n$ are identical with $S_1, \cdots, S_n$, respectively; (ii) for each $i \geqq 1$, R_{n+i} is formed from T_i by replacing each grammatical rule (j, k) of T_i by $(j + m, k + n)$, if $j, k \neq 0$, and retaining the rule $(0, 0)$, if it belonged to T_i;

[3] For suppose that $r_i = 1$. Then T_{r_i} is correlated to S_0. S_0 is thus the only member of $f(a_{i-1}, r_{i-1})$. Hence $(a_{i-1}, t_i) = (0, 0)$, contrary to the assumption that $t_i \neq 0$ for $1 < i < k + 1$.

(iii) $Y_1, \cdots, Y_m$ are $W_1, \cdots, W_m$, respectively; (iv) $Y_{m+1}, \cdots, Y_{m+m'}$ are identical with $V_1, \cdots, V_m$, respectively; (v) R_0 contains S_0 and all grammatical rules formed from those of T_0 as in step (ii). In terms of state diagrams, the diagram for $\bar{G}$ is formed simply by identifying the initial points of the diagrams for G_1 and G_2, which are otherwise kept distinct. $\bar{G}$ is thus an F_3 grammar of $L_1 \cup L_2$, and we know by Theorem 3 that there exists a corresponding unambiguous F_4 grammar for $L_1 \cup L_2$.

It should be noted that although the "periods" can be eliminated from all ambiguous grammars,[4] it is not in general possible to delete occurrences of the identity element from F_4 grammars. If we define F_5 as the set of unambiguous finite state grammars with no occurrences of the identity element as a transition symbol, we have the following theorem.

THEOREM 4. $L(F_4) \supsetneqq L(F_5)$.

The inclusion is obvious, and the language consisting of the two sentences a and $a \wedge b$ is a simple example of a language of $L(F_4)$ which clearly cannot have an F_5 grammar. Thus if we want to have an unambiguous grammar for every finite state language, we must use periods; if we want to avoid the use of periods, we must have ambiguous grammars for certain languages.

CHARACTERIZATION OF FINITE STATE LANGUAGES

Clearly the loops in the state diagram are of particular relevance to characterizing any finite state language. Suppose that, given a finite state grammar G, we define a *cycle* to be a sequence of states $(S_{a_1}, \cdots, S_{a_m})$ such that $m \geqq 2$, $a_1 = a_m$, and for $1 < i < m$, $a_1 \neq a_i \neq 0$. S_{a_1} will be called the *initial state* of this cycle. A *basic* cycle is one whose initial state is S_0. The sequence of cycles $(C_0, \cdots, C_n)$ is a *chain* if C_0 is a basic cycle, each C_i contains the initial state of C_{i+1} $(0 \leqq i < n)$, and no C_j contains the initial state of $C_{j-i} (1 \leqq i \leqq j \leqq n)$. It is a *completed chain* if no sequence $(C_0, \cdots, C_{n+1})$ is a chain. Clearly G has only a finite number of cycles and a finite number of chains. We can use these chains as the basis for constructing a grammar of a particularly simple structure equivalent to G.

Construct the grammar G^* in the following manner. Suppose that the

[4] This is a trivial consequence of Theorem 2. We assume here that W_0 alone represents no sentence; i.e., that a sequence of states generates a sentence only if at least one transition symbol is not the identity symbol.

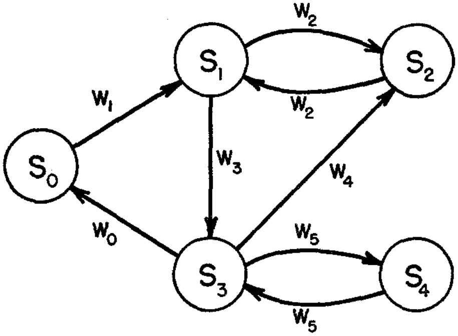

FIG. 1

completed chains of G are $H_1, \cdots, H_p$. Construct states $T_1, \cdots, T_p$, where the sole member of each T_i is $(0, i)$. These "absorbing states" will be used (in the manner of p. 94, above) to index and distinguish occurrences of states in different chains. Next, set an arbitrary one-one correspondence between finite sequences of integers and integers greater than p. This will be used for purely notational purposes. If $(b_1, \cdots, b_k) \leftrightarrow b$ under this correspondence, we will use $(b_1, \cdots, b_k)$ and b interchangeably in characterizing states and grammatical rules.

Suppose now that $H_i = (C_0, \cdots, C_m)$ is the ith completed chain of G, where $(a_0, \cdots, a_m)$ is the sequence of indices of the initial states of $C_0, \cdots, C_m$, respectively. Suppose that $C_j = (S_{j_1}, \cdots, S_{j_n})$ $(0 \leqq j \leqq m)$. For $1 \leqq k < n$, construct the state $T_{(i,a_0,\cdots,a_j,j_k)}$ containing $(0, i)$ and each grammatical rule $[r, (i, a_0, \cdots, a_j, j_{k+1})]$ such that $(r, j_{k+1}) \in S_{j_k}$.

Having carried out this construction for each chain, identify the states labeled $T_{(x,a_0,\cdots,a_j,a_j)}$ and $T_{(y,a_0,\cdots,a_j)}$.[5] Let $T_{(x,0)}$ be the initial state of G^*.

The effect of this construction can best be made clear by an example. Consider the finite state grammar G with the state diagram in Fig. 1. The cycles, completed chains, and initial index sequences of G are shown in Table 1. The construction gives the following states for G^*:[6] (T_1, T_2, T_3, T_4 are absorbing states.)

[5] Where $x = y$, this means that $T_{(x,a_0,\ldots,a_j)}$ is identified with $T_{(x,a_0,\ldots,a_j,j_1)}$, since $j_1 = a_j$. Where $x \neq y$, it means essentially that one state heads several subchains. Equivalently, we could have incorporated this condition into the definition of the correspondence between sequences of integers and integers.

[6] We omit commas in giving the numerical indices; that is, T_{101} stands for $T_{(1,0,1)}$, (1, 101) stands for [1, (1, 0, 1)], etc.

$T_{100} = \{(0, 1), (1, 101)\}$
$T_{101} = \{(0, 1), (3, 103)\}$
$T_{103} = \{(0, 1), (0, 100)\}$
$T_{1011} = \{(0, 1), (2, 1012\}$
$T_{1012} = \{(0, 1), (2, 1011)\}$
$T_{200} = \{(0, 2), (1, 201)\}$
$T_{201} = \{(0, 2), (3, 203)\}$
$T_{203} = \{(0, 2), (0, 200)\}$
$T_{2011} = \{(0, 2), (3, 2013)\}$
$T_{2013} = \{(0, 2), (4, 2012)\}$
$T_{2012} = \{(0, 2), (2,2011)\}$
$T_{20133} = \{(0, 2), (5, 20134)\}$
$T_{20134} = \{(0, 2), (5, 20133)\}$
$T_{300} = \{(0, 3), (1, 301)\}$
$T_{301} = \{(0, 3), (3, 303)\}$
$T_{303} = \{(0, 3), (0, 300)\}$
$T_{3033} = \{(0, 3), (4, 3032)\}$
$T_{3032} = \{(0, 3), (2, 3031)\}$
$T_{3031} = \{(0, 3), (3, 3033)\}$
$T_{30322} = \{(0, 3), (2, 30321)\}$
$T_{30321} = \{(0, 3), (2,30322)\}$
$T_{400} = \{(0, 4), (1, 401)\}$
$T_{401} = \{(0, 4), (3, 403)\}$
$T_{403} = \{(0, 4), (0, 400)\}$
$T_{4033} = \{(0, 4), (5, 4034)\}$
$T_{4034} = \{(0, 4), (5, 4033)\}$

The last step in the construction requires us to identify states in the following manner:

$$T_{10} = T_{100} = T_{200} = T_{300} = T_{400}$$
$$T_{101} = T_{1011} = T_{2011} = T_{201} = T_{301} = T_{401}$$
$$T_{2013} = T_{20133}$$
$$T_{103} = T_{3033} = T_{203} = T_{303} = T_{403} = T_{4033}$$
$$T_{3032} = T_{30322}$$

We thus construct the state diagram for G^* in Fig. 2, where T_{10} is the initial state. (The absorbing states $T_1, \cdots, T_4$ are omitted in this figure.

In G^* the analysis of G into chains is made explicit. Clearly in this

TABLE I

Cycles	Completed chains	↔	Corresponding initial index sequences
$C_0 : (S_0, S_1, S_3, S_0)$	$\mathrm{H}_1 : (C_0, C_1)$	↔	(0, 1)
$C_1 : (S_1, S_2, S_1)$			
$C_2 : (S_1, S_3, S_2, S_1)$	$H_2 : (C_0, C_2, C_6)$	↔	(0, 1, 3)
$C_3 : (S_2, S_1, S_2)$	$H_3 : (C_0, C_5, C_3)$	↔	(0, 3, 2)
$C_4 : (S_2, S_1, S_3, S_2)$	$H_4 : (C_0, C_6)$	↔	(0, 3)
$C_5 : (S_3, S_2, S_1, S_3)$			
$C_6 : (S_3, S_4, S_3)$			
$C_7 : (S_4, S_3, S_4)$			

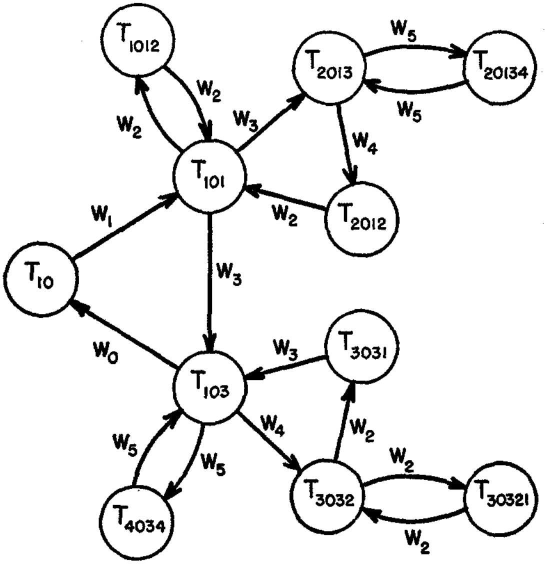

FIG. 2

example G and G^* are equivalent. That this is so in general follows from the observation that

$$\text{if } T_r \text{ is associated with } S_j,\ (i, j) \in S_k, \text{ and the sequence } (S_k, S_j) \text{ occurs in the generation of some sentence, then there is a } T_s \text{ associated with } S_k \text{ such that } (i, r) \in T_s, \tag{1}$$

where the states of G^* associated with a given state S_j of G are those with indices ending in j. Remark (1) follows from the fact that some permutation of each cycle of G belongs to some chain of G, if this cycle appears in the generation of some sentence. Hence (S_k, S_j) is a part of some cycle which belongs to a chain, and the construction outlined above will have associated an appropriate T_s and T_r with S_k and S_j. From (1) it follows, by induction on the length of the generating sequence of states, that $L_{G^*} \supset L_G$. The converse is obvious from the construction. Hence $L_G = L_{G^*}$. More generally, if we let F_6 be the set of grammars constructed in the manner described above, then we have the following theorem.

THEOREM 5. $L(F_6) = L(F_1)$.

Suppose that we recursively define the notation

$$a_1(a_2, \cdots, a_m)a_{m+1} \tag{2}$$

where the a_i's are strings or again where they are notations of the form

$$x_1(x_2, \cdots, x_t)x_{t+1}, \tag{3}$$

etc., in the following way. Let Q_1 be the set of all sequences of the form $(b_1, \cdots, b_{n+1})$, where $b_1 = a_1$, $b_{n+1} = a_{m+1}$, and each b_i $(2 \leqq i \leqq n)$ is one of $a_2, \cdots, a_m$. Let Q_2 be the set of sequences formed from the sequences of Q_1 by expanding the b_i's which are not already strings in the same manner, etc. Then for some r, Q_r will be a set of sequences $(z_1, \cdots, z_s)$, where each z_i is a string. Each such sequence represents the string $z_1 \wedge \cdots \wedge z_s$, and (2) represents the set of these strings.

It is clear that for any $G^* \in F_6$, L_{G^*} can be represented completely in a finite manner in this notation. In fact, the representation can be read off directly from the state diagram. In the example discussed above, L_{G^*} is characterized by the single representation

$$W_1(W_2 \wedge W_2, W_3(W_5 \wedge W_5)W_4 \wedge W_2)W_3(W_5 \wedge W_5, \\ W_4(W_2 \wedge W_2)W_2 \wedge W_3)W_0. \tag{4}$$[7]

By virtue of Theorem 5, we have the following result.

THEOREM 6. *Any finite state language can be represented by a finite number of finite notations of the form* (2).

[7] If there were more than one transition possible between a given pair of distinct states in the original G, there would be more than one such representation. This would also be true if there were more than one basic cycle.

Notice that this characterization of finite state languages can be made more economical. If in the grammar pictured in Fig. 2 the states T_{3032}, T_{30321} and T_{3031} are deleted, exactly the same language will be generated. Correspondingly, in (4) we can delete the subrepresentation $W_4(W_2 \wedge W_2)W_2 \wedge W_3$ without altering the represented language. The reason for this is that the chains listed in Table I are in a certain sense a redundant set. Thus H_3 never figures in the generation of any sentence that is not already generated by virtue of the other chains. Suppose that we say that a chain $(C_0, \ldots, C_m, C_{b_1}, \ldots, C_{b_n})$ is a *redundant* chain of G if there is a chain $(C_0, \ldots, C_m, C_a)$ of G such that C_{b_1} is, in the obvious sense, a cyclic permutation of C_a and the initial state of C_a precedes the initial state of C_{b_1} in C_m. Then H_3 is a redundant chain in the case considered. Suppose now that in constructing the grammar G^* as above we consider only the completed chains of G which are not redundant. This, in fact, effects no limitation on the generative power of G^* and it leads to a more economical representation.

Notice that the converse of Theorem 6 is obviously true, so that finite representability in terms of the notation (2) is a necessary and sufficient characterization of finite state languages.

THE ALGEBRA OF FINITE STATE LANGUAGES

We have seen (corollary to Theorem 3) that the union of any two languages with F_4 grammars is again a language with an F_4 grammar. Recalling that, throughout, we are considering a language to be some subset of the set U of all finite strings in some given vocabulary (see above, p. 94), we can now ask whether the complement of a language with an F_4 grammar (i.e., the set of strings of U not contained in the given language) is a language with an F_4 grammar. This is in fact the case.

THEOREM 7. *If* $G \in F_4$, *then there is a* $G^* \in F_4$ *such that* L_{G^*} *is the complement of* L_G *in* U.

Suppose that the vocabulary of U is $W_1, \cdots, W_m$, and that G contains the states $S_0, \cdots, S_n$. We construct G^* containing states $T_0, \cdots, T_{n+1}$ as follows:

(i) if $i \neq 0$ and $(i, k) \in S_j$, then $(i, k) \in T_j$

(ii) If $i \neq 0$ and there is no k such that $(i, k) \in S_j$, then $(i, n + 1) \in T_j$

(iii) $(0, 0) \in T_{n+1}$ and $(i, n + 1) \in T_{n+1}$ for $1 \leqq i \leqq m$

(iv) $(0, 0) \in T_j$ if and only if $(0, 0) \notin S_j$

(v) nothing else is included in T_j.

Definition (i) has the effect of reproducing in G^* all paths which, with the addition of W_0, lead to sentences in G. Exactly this set of strings is excluded from L_{G^*} by the stipulation in (iv) that no transition to S_0 in G is a transition to T_0 in G^*. By (ii) we add to T_j all transition symbols that did not appear in S_j, but we cause them to lead to a new "universal" state T_{n+1} which has no counterpart in G. Definition (iii) provides that all strings are possible once we reach T_{n+1}, including termination by W_0. Definition (iv) insures that every state which does not lead to S_0 will lead to T_0 in G^*.

G was by assumption an unambiguous grammar, and clearly no ambiguity was introduced into G^* by the construction. We can therefore be certain that no sentence excluded from L_{G^*} by the "only if" condition of (iv), can be reintroduced by any of the other steps of the construction. Consequently, L_{G^*} is exactly the complement of L_G.

Notice that $(G^*)^*$ is identical with G except for transitions to T_{n+1}.

But T_{n+1} is now a vacuous state, since (0, 0) will have been excluded from it by the construction. Hence $(G^*)^*$ is in fact a grammar of L_G.

From Theorem 7 and the corollary to Theorem 3 it follows immediately that the set of languages with F_4 grammars forms a Boolean algebra of sets with universal element U. Combining this result with Theorems 1, 2, and 3, we have

THEOREM 8. *The set of finite state languages in a fixed vocabulary forms a Boolean algebra of sets with U as universal element, where U is the set of all finite strings in this vocabulary.*

THE NUMBER OF SENTENCES OF LENGTH λ

When we use a finite state language we often encounter the practical problem of determining how many sentences it contains. Except for the trivial case in which the grammar has no loops, the answer to this question is always the same: The language contains an infinite number of different sentences. Nevertheless, there is a sense in which one language contains more sentences than another; if we consider how the number of different sentences increases as their length increases, one language may grow much more rapidly than another. What we are interested in, therefore, is what Mandelbrot (1955) has called the "structure function" of the language. The structure function $f(\lambda)$ is the number of sentences of exactly length λ. By means of the structure function we can compare the size of any two languages for any given value of λ. Closely related to the structure function is the total number of sentences of length λ or less, which is given by $f(1) + f(2) + \cdots + f(\lambda)$.

The structure function should not be confused with the number of *messages* of any given length that can be formed in a given language. A message may be composed of a string of sentences, so the number of messages of any given length is usually much larger than the number of sentences of that length. Shannon (1948) has presented an example of the method used to compute the number of different messages of a given length for the special case of telegraph symbols. He established the general result that for finite state channels the number of different messages increases exponentially as a function of length. This fact leads quite directly to a definition of channel capacity which we shall adopt below in Eq. (7). The same exponential rate of growth also holds for the structure function, although the number of different sentences usually increases more slowly than the number of different strings of sentences.

Thus it is natural to define another informational capacity, analogous to that defined by Shannon for messages, but with the condition that the message must consist of a single sentence. In the following, therefore, we shall distinguish between the informational capacity "for sentences"—where the entire message consists of a single sentence—and the informational capacity "for messages" in the sense defined by Shannon—where the message may consist of one or more sentences.

These three quantities—the number of sentences of length λ, the number of sentences of length λ or less, and the number of messages of length λ—are each considered in turn in this and the following two sections. In all three sections it will be assumed that we are dealing with an F_4 grammar; that is to say, that the grammar is unambiguous, so that each sentence is generated via a unique path, and that the identity element occurs only terminally. From Theorems 1–5 we know that this assumption does not limit the generality of our results. We turn now to the determination of the structure function.

If the terminal identity element is not considered to contribute to the length of a sentence, then we can write

$$\begin{aligned} f(0) &= f_{00}(1) = 0 \\ f(1) &= f_{00}(2) = \sum_{i=1}^{n} f_{0i}(1) f_{i0}(1) \\ f(\lambda + 1) &= f_{00}(\lambda + 2) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{0i}(1) f_{ij}(\lambda) f_{j0}(1), \quad \lambda = 1, 2, \cdots \end{aligned} \tag{5}$$

where $f_{ij}(\lambda)$ is the number of paths from state i to state j that involve exactly λ transitions. If the set of $f_{ij}(\lambda)$ for which $i, j \neq 0$ are considered to be elements of an n-by-n matrix M_λ, where $M_0 = I$, then Eq. (5) can be written

$$f(\lambda + 1) = vM_\lambda w, \qquad \lambda = 0, 1, 2, \cdots$$

where v is the row matrix $[f_{01}(1), \cdots, f_{0n}(1)]$ and w is the transpose of the row matrix $[f_{10}(1), \cdots, f_{n0}(1)]$. Since

$$f_{hj}(\lambda + 1) = \sum_{i=1}^{n} f_{hi}(\lambda) f_{ij}(1),$$

we see that $M_{\lambda+1} = M_\lambda M_1$ and by induction we obtain $M_\lambda = M_1^{\lambda}$. Thus we have

$$f(\lambda + 1) = vM_1^{\lambda} w, \qquad \lambda = 0, 1, 2, \cdots .$$

The matrix M_1 can be constructed directly from the grammar or the state diagram by counting the number of transitions between states, ignoring those transitions into and out of state 0 which are considered in order to construct v and w.

By the Cayley-Hamilton theorem,

$$M_1^{\lambda} = -c_{n-1}M_1^{\lambda-1} - \cdots - c_0 M_1^{\lambda-n}$$

where the constants c_i are given by the characteristic equation

$$| M_1 - rI | = 0.$$

Since $M_1^{\lambda} = M_\lambda$, we have

$$M_\lambda = -c_{n-1}M_{\lambda-1} - \cdots - c_0 M_{\lambda-n}.$$

Because this obviously applies to each element of M_λ, we can write

$$f_{ij}(\lambda) = -c_{n-1}f_{ij}(\lambda - 1) - \cdots - c_0 f_{ij}(\lambda - n),$$

$$\lambda = n + 1, n + 2, \cdots.$$

Substituting this expression into the original Eq. (5) for $f(\lambda + 1)$ gives

$$\begin{aligned} f(\lambda + 1) &= \sum_{j=1}^{n} \sum_{i=1}^{n} f_{0i}(1)[- c_{n-1}f_{ij}(\lambda - 1) - \cdots - c_0 f(\lambda - n)]f_{j0}(1) \\ &= -c_{n-1}f(\lambda) - \cdots - c_0 f(\lambda - n + 1). \end{aligned} \tag{6}$$

In this way we obtain a finite difference equation for the number of sentences of length λ; given $f(1), \cdots, f(n)$, we can compute $f(\lambda)$ recursively for values of $\lambda > n$.

Thus we see that the characteristic equation of M_1, together with $f(1), \cdots, f(n)$, can be solved in order to obtain a general expression for $f(\lambda)$. The characteristic equation of M_1 can be written

$$| M - rI | = r^n + c_{n-1}r^{n-1} + \cdots + c_1 r + c_0 = 0.$$

If we let $\mathrm{sp}_k M$ represent the sum of the principal minors of M_1 of order k, the equation becomes

$$r^n - \mathrm{sp}_1 M r^{n-1} + \mathrm{sp}_2 M r^{n-2} - \cdots \pm \mathrm{sp}_{n-1} M r \mp | M | = 0.$$

The principal minors are closely related to the cycles of the grammar. Thus $\mathrm{sp}_1 M$ is the sum of the elements f_{ii}, which are the loops of length one; therefore, $\mathrm{sp}_1 M$ is the total number of loops of length one. Similarly, $\mathrm{sp}_2 M$ is the sum of all 2×2 determinants of the form $f_{ii}f_{jj} - f_{ij}f_{ji}$, which involves loops of length one and length two. It is obvious that if the grammar contains no cycles, all principal minors will be zero, and there will be no sentences longer than n. A grammar that contains exactly one loop of length k will have exactly one nonzero principal minor, $f_{i_1 i_2} \cdots f_{i_k i_1} = (-1)^{k+1}$. Thus the characteristic equation becomes $r^k - 1 = 0$, and $f(\lambda)$ will equal one each time λ increases by k (unless state i_1 is an "absorbing" state). As we would expect intuitively from a study of the state diagrams, the only interesting cases arise when the grammar contains at least two loops.

Frobenius established that for matrices consisting of nonnegative real elements, the dominant root is positive. Moreover, when two matrices M and N having only nonnegative real elements f_{ij} and g_{ij} are so related that $f_{ij} \geqq g_{ij}$, then the dominant root of M will be equal to or larger than the dominant root of N. If we add a loop to a grammar which contains only one, therefore, the result will never reduce the dominant root below the roots of unity. In fact, if any basic cycle in the grammar has two or more loops attached to it, the largest real root will be greater than unity.

The general solution of Eq. (6) is well known to be

$$f(\lambda) = a_1 r_1^{\lambda} + a_2 r_2^{\lambda} + \cdots + a_n r_n^{\lambda}.$$

Assume $r_1 > 1$ is the dominant root, so as $\lambda \to \infty$,

$$\lim_{\lambda \to \infty} f(\lambda) = a_1 r_1^{\lambda}, \qquad \text{or} \qquad \lim_{\lambda \to \infty} \frac{\log f(\lambda)}{\lambda} = \log r_1 . \tag{7}$$

Eq. (7) has been used by Shannon (1948) in a slightly different context as a definition of channel capacity. By analogy, it can be defined here as the informational capacity of sentences, and if logarithms to the base 2 are used, Eq. (7) gives us the capacity in bits per word.

THE NUMBER OF SENTENCES OF LENGTH λ OR LESS

Now suppose we consider any finite state language L to be the union of a denumerable set of disjoint finite state languages L_λ, $\lambda = 1, 2, \cdots$, where L_λ contains only sentences of L of length λ. This is permissible, since each L_λ contains a finite number $f(\lambda) = f(L_\lambda)$ of different sentences and these can be generated by a finite state grammar. Then f is a measure, since

$$f(L) = f\left(\bigcup_{\lambda=1}^{\infty} L_\lambda\right) = \sum_{\lambda=1}^{\infty} f(L_\lambda).$$

In the interesting cases, however, $f(L)$ is not finite.

The sum of $f(\lambda)$, $\lambda = 1, 2, \cdots, \nu$, is the number of sentences of length ν or less. Define $F(\nu)$ as this sum:

$$\begin{aligned} F(\nu) &= \sum_{\lambda=1}^{\nu} f(\lambda) \\ &= \sum (a_1 r_1^{\lambda} + \cdots + a_n r_n^{\lambda}) \\ &= a_1 \sum r_1^{\lambda} + \cdots + a_n \sum r_n^{\lambda} \\ &= a_1 r_1 \frac{r_1^{\nu} - 1}{r_1 - 1} + \cdots + a_n r_n \frac{r_n^{\nu} - 1}{r_n - 1} \\ &= r_1^{\nu} \left(a_1 \frac{r_1}{r_1 - 1} \left(1 - \frac{1}{r^{\nu}}\right) + \cdots + a_n \frac{r_n}{r_n - 1} \frac{r_n^{\nu} - 1}{r_1^{\nu}} \right). \end{aligned}$$

Thus if $r_1 > 1$ is the dominant root,

$$\lim_{\nu \to \infty} F(\nu) = \frac{r_1}{r_1 - 1} a_1 r_1^{\nu} = \frac{r_1}{r_1 - 1} \lim_{\lambda \to \infty} f(\lambda).$$

Therefore, the dominant root of the characteristic equation for $F(\nu)$ is the same as for $f(\lambda)$, and $\log_2 r_1$, the informational capacity of sentences measured in bits per word, remains unchanged.

THE NUMBER OF STRINGS OF SENTENCES

The number of different strings of complete sentences of length λ can be computed recursively by

$$G(\lambda) = f(\lambda + 1) + \sum_{i=1}^{\lambda} G(\lambda - i) f(i + 1)$$

or by returning to Eq. (5) and noting that the summations now include values of $i, j = 0$, since one sentence can be followed by another. If we assume that the terminal identity element now occupies one unit of length in order to signal the boundaries between successive sentences, we can write

$$G(\lambda + 2) = \sum_{j=0}^{n} \sum_{i=0}^{n} f_{0i}(1) f_{ij}(\lambda) f_{j0}(1).$$

Therefore, we proceed as before, but with a square matrix N_λ of $(n + 1)^2$ elements, where

$$N_1 = \begin{Bmatrix} 0 & v \\ w & M_1 \end{Bmatrix}.$$

The dominant root of N_1 will generally be larger than the dominant root of M_1. If we refer to a string of sentences as a message, then the logarithm of the dominant root of N_1 can be defined as the informational capacity of messages, measured in bits per word, and this will generally be larger than the informational capacity of sentences.

THE NUMBER OF SENTENCES IN THE COMPLEMENTARY LANGUAGE

For the universal language U, which contains all strings that can be formed on a vocabulary of m words, $f(U_\lambda) = m^\lambda$. If for some other language L formed on this same vocabulary, $r_1 < m$, then $f(L_\lambda)/f(U_\lambda) = a_1(r_1/m)^\lambda \to 0$ as $\lambda \to \infty$. The language L^*, which is the complement of L, must satisfy the relation $f(L_\lambda) + f(L_\lambda^*) = f(U_\lambda)$, so it follows that $f(L_\lambda^*)/f(U_\lambda) \to 1$ as $\lambda \to \infty$. This implies that m is the dominant root of the characteristic polynomial of L^*. In other words, for any finite state language L formed on a vocabulary of m words, it must be true that either L or its complement L^* has m as the dominant root of its characteristic polynomial. Therefore, for any sufficiently long string of words chosen at random, the probability that it will form a sentence is either 0 or 1.

RECEIVED: January 24, 1958.

REFERENCES

CHOMSKY, N. (1956). Three models for the description of language. *IRE Trans. on Inform. Theory* **IT-2**, 113–124.

MANDELBROT, B. (1954). On recurrent noise limiting coding. *Proc. Symposium on Information Networks, Polytechnic Institute of Brooklyn*, pp. 205–221.

SCHÜTZENBERGER, M. P. (1956). On an application of semi-group methods to some problems in coding. *IRE Trans. on Inform. Theory* **IT-2**, 47–60.

SHANNON, C. E. (1948). A mathematical theory of communication. *Bell System Tech. J.* **27**, 379–423.