STUDIES IN LOGIC

AND THE

FOUNDATIONS OF MATHEMATICS

L. E. J. BROUWER / E. W. BETH / A. HEYTING

EDITORS

Computer Programming and Formal Systems

Editors

P. BRAFFORT

and

D. HIRSCHBERG

NORTH-HOLLAND PUBLISHING COMPANY

AMSTERDAM

THE ALGEBRAIC THEORY OF CONTEXT-FREE LANGUAGES*

N. CHOMSKY
Massachusetts Institute of Technology

AND

M. P. SCHÜTZENBERGER
Harvard University

1. Linguistic Motivation

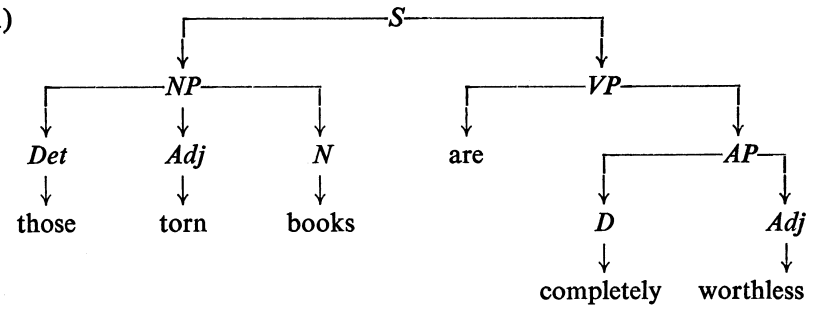
We will be concerned here with several classes of sentence-generating devices that are closely related, in various ways, to the grammars of both natural languages and artificial languages of various kinds. By a *language* we will mean simply a set of strings in some finite set V of symbols called the *vocabulary* of the language. By a *grammar* we mean a set of rules that give a recursive enumeration of the strings belonging to the language. We will say that the grammar *generates* these strings. (Thinking of natural languages, we would call the generated strings *sentences*; in algebraic parlance they would ordinarily be called *words* and the vocabulary would be called an *alphabet*; regarding a grammar as specifying a programming language, the strings would be called *programs*; we will generally use the neutral term *strings*).

For a class of grammars to have linguistic interest, there must be a procedure that assigns to any pair (σ, G), where σ is a string and G a grammar of this class, a satisfactory *structural description* of the string σ with respect to the grammar G. In particular, the structural description should indicate that the string σ is a well-formed sentence of the language $L(G)$ generated by G, where this is the case. If it is, the structural description should contain grammatical information that provides the basis for explaining how σ is understood by speakers who have internalized the grammar G; if it is not, the structural description might indicate in what respects σ deviates from well-formedness.

* This work was supported in part by the U. S. Army Signal Corps, the Air Force Office of Scientific Research, and the Office of Naval Research; and in part by the National Science Foundation; and in part by a grant from the Commonwealth Fund.

We will be concerned with only one aspect of the structural description of a sentence, namely, its subdivision into phrases belonging to various categories. Thus, for example, a structural description of the English sentence "those torn books are completely worthless" should indicate that *those* is a *Determiner*, *torn* and *worthless* are *Adjectives*, *books* is a *Noun*, *completely* is an *Adverb*, *those torn books* is a *Noun Phrase*, *completely worthless* is an *Adjective Phrase*, *are completely worthless* is a *Verb Phrase*, the whole string is a *Sentence*, as well as additional details regarding subclassification. This information can be represented by a diagram such as (1):

(1)



or, equivalently, by a labelled bracketing of the string, as in (2):

(2) $[_S\ [_{NP}\ [_{Det}$ those$][_{Adj}$ torn$][_N$ books$]][_{VP}$ are $[_{AP}\ [_D$ completely$]$ $[_{Adj}$ worthless$]\ ]\ ]$.

A major concern of the general theory of natural languages is to define the class of possible strings (by fixing a universal phonetic alphabet); the class of possible grammars; the class of possible structural descriptions; a procedure for assigning structural descriptions to sentences, given a grammar; and to do all of this in such a way that the structural description assigned to a sentence by the grammar of a natural language will provide the basis for explaining how a speaker of this language would understand this sentence (assuming no limitations of memory, attention, etc.). The grammar, then will represent certain aspects of the linguistic competence of the speaker of the language.

We will not be concerned here with the empirical question of adequacy of the structural descriptions or the grammars that we will investigate. In fact, the classes of grammars that we will consider, and the kinds of

structural descriptions that they generate, are undoubtedly too narrow to do justice to real human linguistic competence. Nevertheless, the systems we consider (which, in effect, formalize traditional notions of parsing and immediate constituent analysis) bear certain relations to the kinds of systems that seem empirically adequate, and that are, for the time being, too complex to permit abstract study.[1])

In the representation (2), we have, aside from brackets, two kinds of symbols: (i) symbols of the generated string (i.e., the six symbols *those, torn, books, are, completely, worthless*)[2]); (ii) the symbols *S, NP, Det, Adj, N, VP, AP, D* representing phrase-categories. Symbols of type (i) we will call *terminals;* symbols of type (ii), *non-terminals.*

We will assume, below, a fixed stock of terminal and non-terminal symbols from which the grammars of all languages are constructed. The set of terminals can be regarded as constituting a potential common vocabulary for all languages. Thinking of spoken language, we can regard the set of terminals as defined by a universal phonetic alphabet (assuming, as is natural, an upper bound on the length of morphemes[2]). Thinking again of natural language, we can regard the fixed set of non-terminals as a universal set of categories from which the phrase types of all languages are drawn. An important and traditional question of general linguistics has to do with the possibility of giving a concrete interpretation of the non-terminals that constitute the categories in terms of which grammars are constructed — is it possible, in other words, to find a general definition, independent of any particular language, of such categories as Noun, Verb, etc., in terms of semantic content or formal properties of grammars? The problem of giving a concrete interpretation to the set of terminals and non-terminals is, of course, like the problem of empirical adequacy of certain categories of grammars, a crucial issue in the science of language; but it is beyond the range of our immediate interests here.

We can generate the sentence "those torn books are completely worthless", with the structural description (2), by the set of *rewriting rules*:

---

[1]) For further discussion of these questions, see Chomsky [10].

[2]) In a linguistically adequate grammar, we would generate not these symbols, but rather a more abstract representation using the symbols *the, demonstrative, plural, tear, participle, book, plural, be, plural, complete, ly, worth, less,* in this order. Representation in terms of these symbols (called *morphemes*) will be converted to phonetic representation by a set of *phonological rules* which will not concern us at all here. See Chomsky and Miller [15]. We will use actual sentences, such as (2), only for illustrative examples, and will therefore not be concerned with such refinements as this.

(3)
$$\begin{aligned} S &\to NP\ VP \\ NP &\to Det\ Adj\ N \\ Det &\to \text{those} \\ Adj &\to \text{torn} \\ Adj &\to \text{worthless} \\ N &\to \text{books} \\ VP &\to \text{are}\ AP \\ AP &\to D\ Adj \\ D &\to \text{completely} \end{aligned}$$

by a *derivation* that is constructed in the following way. First, write down the *initial symbol* S as the first line of the derivation. Form the $n + 1^{st}$ line of the derivation by selecting at will an occurrence of a non-terminal α in the n^{th} line (where this occurrence of α is not labelling a bracket), and replace it by the string: $[_\alpha \varphi]$, where $\alpha \to \varphi$ is one of the rules of (3). Continue until the only non-terminals that appear are those that label brackets, at which point, the derivation is *terminated.* Deleting the brackets of a terminated derivation, with their labels, we have a string containing only terminals. Call this a *terminal string.* Four different terminal strings can be generated by the grammar (3). We can construct a grammar that generates infinitely many terminal strings, each with a structural description, by permitting recursions, e.g., by adding to (3) the rules

(4)
$$\begin{aligned} NP &\to \text{that}\ S \\ VP &\to \text{is}\ AP \\ AP &\to \text{obvious} \end{aligned}$$

in which case we can generate, e.g., "that those torn books are completely worthless is obvious", etc.[1]). Each of the generated sentences will again have a structural description of the appropriate kind.

Grammars of the type (3), (4) we will call *context-free (CF) grammars.* They are characterized by the property that exactly one non-terminal appears on the left-hand side of each rewriting rule. If this restriction is relaxed, we have systems with entirely different formal properties. It

[1]) In this case, infinitely many non-English sentences will also be generated, e.g., "that those torn books is obvious are completely worthless", etc. Hence the grammar ((3), (4)) is unacceptable. The difficulty of avoiding empirical inadequacies of this sort can easily be underestimated. We stress again that this is the key issue for both linguistics and psychology, though it will not concern us directly here. For discussion, see Chomsky [13].

seems that grammars for natural languages must contain at least some rewriting ryles of this more general form, and some rules that are not rewriting rules at all. Cf. Chomsky [8], [10], and [12], Chomsky and Miller [15], for further abstract discussion of systems of these sorts, which we will not consider further here. A set of terminal strings that can be generated by some *CF* grammar we will call a *CF language.*

A *CF* language may generate a terminal *(debracketised)* string φ with several different structural descriptions. In this case, if the grammar is empirically adequate, φ should be structurally ambiguous. Consider, for example, the *CF* grammar with the rules

(5)
$S \rightarrow NP\ VP$
$NP \rightarrow$ they; $NP \rightarrow Adj\ N$; $NP \rightarrow N$
$VP \rightarrow$ are NP; $VP \rightarrow$ Verb NP
Verb $\rightarrow$ are flying
$Adj \rightarrow$ flying
$N \rightarrow$ planes

With this grammar we can generate both (6) and (7):

(6) $[_S[_{NP}$ they$]\ [_{VP}[_{Verb}$ are flying$]\ [_{NP}[_N$ planes$]\ ]\ ]\ ]$.
(7) $[_S[_{NP}$ they$]\ [_{VP}$ are $[_{NP}\ [_{Adj}$ flying$]\ [_N$ planes$]\ ]\ ]\ ]$

Correspondingly, the terminal string "they are flying planes" is structurally ambiguous; it can mean: "my friends, who are pilots, are flying planes"; or: "those spots on the horizon are flying planes". Study of structural ambiguity is one of the most instructive ways to determine the empirical adequacy of a grammar.

We will see below that there are certain *CF* languages that are inherently ambiguous, in the sense that each *CF* grammar that generates them assigns alternative structural descriptions to some of their sentences. Furthermore, we will see that the problem of determining whether a *CF* grammar is ambiguous is recursively unsolvable,[1]) even for extremely simple types of *CF* grammars.

Though *CF* grammars are far from fully sufficient for natural languages, they are certainly adequate for the description of familiar artificial languages, and apparently for the description of certain, perhaps all, programming languages. In particular, a *CF* grammar can be written for

[1]) There is, in other words, no mechanical procedure (algorithm) for determining whether an arbitrary *CF* grammar assigns more than one structural description to some string that it generates.

ALGOL [18], and each program in ALGOL will be one of the terminal strings generated by this grammar. Clearly, a programming language must be unambiguous. Therefore, it is important to determine whether, in fact, a particular programming language meets this condition, or whether a particular infinite set of programs *can* each be unambiguous, given certain techniques for constructing them (e.g., techniques that can be represented as rules for constructing derivations in a *CF* grammar). As indicated in the preceding paragraph, these may be rather difficult questions.

Suppose that G_1 and G_2 are generative systems that specify certain techniques for constructing computer programs; suppose, in fact, that they are grammars that generate the programming languages L_1 and L_2, each of which consists of an infinite number of strings, each string being a possible program. It is often interesting to inquire into the relative power of programming languages. We will see that if G_1 and G_2 are *CF* grammars (as, e.g., in the case of ALGOL), most problems concerning the relation of L_1 to L_2 are recursively unsolvable, in particular, the problem of determining whether L_1 and L_2 have an empty or an infinite intersection, or whether L_1 is contained in L_2 [2], or whether there is a finite transducer (a "compiler") that maps L_1 onto L_2 (Ginsburg and Rose, personal communication). Hence it is possible that general questions concerning the formal properties of *CF* systems and formal relations between them may have a concrete interpretation in the study of data-processing systems, as well as in the study of natural language. This possibility has been pointed out particularly by Ginsburg and Rice [18], Ginsburg and Rose [19].

In considering a grammar as a generative device, we may be concerned with the language (i.e., set of terminal strings) that it generates, or with the set of structural descriptions that it generates (N.B.: each structural description uniquely determines a terminal string, as in (2)). The latter is clearly the much more interesting question. Similarly, in studying generative capacity of a class of grammars (or relative capacity of several such classes, as in evaluating alternative linguistic theories), we may be concerned either with the set of languages that can be generated, or with the set of systems of structural descriptions that can be generated. The latter, again, is a more interesting, but much more difficult question. Investigation of such questions is, altogether, quite recent, and attention has been restricted almost exclusively to generation of languages rather than of systems of structural descriptions. We will consider genera-

tion from a point of view intermediate between the two just mentioned. We will consider a representation of a language not as a set of strings and not as a set of structural descriptions, but as a set of pairs (σ, n), where σ is a string and n expresses its degree of ambiguity; that is, n is the number of different structural descriptions assigned to σ by the grammar G generating the language to which it belongs.

2. Grammars as Generators of Formal Power Series

2.1. Suppose that we are given a finite vocabulary V partitioned into the sets V_T (= terminal vocabulary) and V_N (= non-terminal vocabulary). We consider now languages with the vocabulary V_T, and grammars that take their non-terminals from V_N. Let $F(V_T)$ be the free monoid generated by V_T, i.e., the set of all strings in the vocabulary V_T. A language is, then, a subset of $F(V_T)$.

Consider a mapping r which assigns to each string $f \in F(V_T)$ a certain integer $\langle r, f\rangle$. Such a mapping can be represented by a *formal power series* (denoted also by r) in the non-commutative variables x of V_T. Thus

$$r = \sum_i \langle r, f_i\rangle f_i = \langle r, f_1\rangle f_1 + \langle r, f_2\rangle f_2 + \ldots, \tag{8}$$

where $f_1, f_2, \ldots$ is an enumeration of all strings in V_T. We define the support of r (= Sup(r)) as the set of strings with non-zero coefficients in r. Thus

$$\mathrm{Sup}(r) = \{f_i \in F(V_T) \mid \langle r, f_i\rangle \neq 0\}. \tag{9}$$

We do not insist that the coefficients $\langle r, f_i\rangle$ of the formal power series r in (8) be positive. If, in fact, for each i, $\langle r, f_i\rangle \geq 0$, then we shall say that r is a *positive* formal power series.

If for each $f_i \in F(V_T)$, the coefficient $\langle r, f_i\rangle$ is either zero or one, we say that r is the *characteristic* formal power series of its support.

2.2. If r is a formal power series and n an integer, we define the product nr as the formal power series with coefficients $\langle nr, f\rangle = n\langle r, f\rangle$, where $\langle r, f\rangle$ is the coefficient of f in r. Where r and r' are formal power series, we define $r + r'$ as the formal power series with coefficients $\langle r + r', f\rangle = \langle r, f\rangle + \langle r', f\rangle$, where $\langle r, f\rangle$ and $\langle r', f\rangle$ are, respectively, the coefficients of f in r and r'. We define rr' as the formal

power series with coefficients $\langle rr', f\rangle = \sum_{i,j} \langle r, f_i\rangle \langle r', f_j\rangle$, where $f_i f_j = f$. Thus the set of formal power series form a ring closed under the operations: multiplication by an integer, addition, multiplication.

Notice that where r and r' are positive formal power series the support of $r + r'$ is exactly the set union of the supports of r and r', and the support of rr' is exactly the set product of the supports of r and r' (i.e., the set of all strings $f_i f_j$ such that f_i is in the support of r and f_j in the support of r'). We will discuss the interpretation of other simple set theoretic operations below.

We say that two formal power series r and r' are *equivalent mod degree n* (i.e., $r \equiv r'$ (mod deg n)) if $\langle r,f\rangle = \langle r',f\rangle$ for every string f of length ("degree") $\leq n$. Suppose then that we have an infinite sequence of formal power series $r_1, r_2, \ldots$, such that for each n and each $n' > n$, $r_n \equiv r_{n'}$ (mod deg n). In this case, the limit r of the sequence $r_1, r_2, \ldots$. is well-defined as

$$r = \lim_{n \to \infty} \pi_n r_n \tag{10}$$

where for each n, $\pi_n r_n$ is the polynomial formed from r_n by replacing all coefficients of strings of length $> n$ by zero. Then the ring of the formal power series becomes an *ultrametric*, hence topological, ring.

With these notions defined, we can turn to the problem of relating the representation of languages in terms of formal power series to the representation of languages by generative processes such as *CF* grammars.

2.3. Suppose that G is a generative process generating the language $L(G)$. Each string $f \in F(V_T)$ is assigned a certain number $N(G,f)$ of structural descriptions by G; $N(G,f) > 0$ just in case $f \in L(G)$. $N(G,f)$ expresses the degree of structural ambiguity of f with respect to G. It is natural to associate with G the formal power series $r(G)$ such that $\langle r(G),f\rangle = N(G,f)$, where $\langle r(G),f\rangle$ is the coefficient of f in $r(G)$. Thus $r(G)$ expresses the ambiguity of all terminal strings with respect to the grammar G. The coefficient $\langle r(G),f\rangle$ is *zero* just in case f is not generated by G; it is *one* just in case f is generated unambiguously (in one and only one way) by G; it is *two* just in case there are two different structural descriptions for f, in terms of G; etc.

An $r(G)$ associated with a grammar G will, of course, always be positive; and its support Sup($r(G)$) will be exactly the language $L(G)$ generated by G. We can regard a formal power-series r with both positive and negative coefficients as being associated with *two* generative processes

G_1 and G_2. The coefficient $\langle r,f\rangle$ of f in r can be taken as the difference between the number of times that f is generated by G_1 and by G_2; that is, in this case, $\langle r,f\rangle = N(G_1,f) - N(G_2,f)$.

Suppose that G is a *CF* grammar with non-terminals $\alpha_1, \ldots, \alpha_n$, where α_1 is the designated initial symbol (i.e., $\alpha_1 = S$, in the example (1), above). We can construct the formal power series $r(G)$ associated with G by a straightforward iterative procedure. To do this, we proceed as follows.

Observe, first of all, that G can be written as a system of equations in the variables $\alpha_1, \ldots, \alpha_n$. Let $\varphi_{i,1}, \ldots, \varphi_{i,m_i}$ be the strings such that $\alpha_i \to \varphi_{i,j}$ $(1 \leq j \leq m_i)$ are rules of G. We then associate with α_i the *polynomial expression* σ_i,

$$\sigma_i = \varphi_{i,1} + \varphi_{i,2} + \ldots + \varphi_{i,m_i} \tag{11}$$

We now associate with the grammar G the set of equations

$$\alpha_1 = \sigma_1; \ldots; \alpha_n = \sigma_n. \tag{12}$$

Let us assume that the grammar G contains no rules of the form

$$\begin{aligned} &\alpha_i \to e \\ &\alpha_i \to \alpha_j. \end{aligned} \tag{13}$$

It is clear that these assumptions do not affect generative capacity [2]. That is, for every *CF* grammar containing such rules there is another grammar without any such rules, which generates the same language. We will also explicitly require, henceforth, that if G is a *CF* grammar and α is a non-terminal of G, then there must be terminal strings derivable from α — i.e., if G' contains the rules of G and has α as its initial symbol, then the language generated by G' must be non-null. Again, this requirement obviously does not affect generative capacity.

Returning now to the problem of constructing the power-series that is associated with G and that represents the degree of ambiguity that G assigns to each string, observe that we can regard each equation $\alpha_i = \sigma_i$ of (12) as defining a mapping ψ_i that carries an n-tuple $(r_1, \ldots, r_n)$ of power series into the power series defined by replacing α_j in σ_i by r_j. This is legitimate because of the closure properties of the ring of power series noted above in § 2.2.

Thus the set of equations (12) defines a mapping ψ,

$$\psi(r_1, \ldots, r_n) = (r'_1, \ldots, r'_n), \text{ where } r'_1 = \psi_i(r_1, \ldots, r_n). \tag{14}$$

Consider now the infinite sequence of n-tuples of power series ϱ_0, ϱ_1, ..., where

$$\begin{aligned} \varrho_0 &= (r_{0,1}, \ldots, r_{0,n}) = (0, \ldots, 0) \\ \varrho_1 &= (r_{1,1}, \ldots, r_{1,n}) \\ \varrho_2 &= (r_{2,1}, \ldots, r_{2,n}) \end{aligned} \tag{15}$$

and where for each i,j $(j > 0)$

$$r_{j,i} = \psi_i (r_{j-1,1}, \ldots, r_{j-1,n}), \tag{16}$$

and where 0 is the power series in which all coefficients are zero. Each $r_{j,i}$ in (15) has only finitely many non-zero coefficients; it is, in other words, a polynomial. Furthermore, we can show that for each i,j,j' such that $j' > j > 0$, $1 \leq i \leq n$, it is the case that

$$r_{j,i} \equiv r_{j',i} \pmod{\deg j}. \tag{17}$$

Consequently, as noted in § 2,2, the limit $r_{\infty,j}$ of the infinite sequence $r_{1,i}$, $r_{2,i}$, ... is well-defined for each i (it is, of course, in general not a polynomial). We will call the n-tuple $(r_{\infty,1}, \ldots, r_{\infty,n})$, so defined, *the solution* to the set of equations (12). Indeed, the n-tuple $(r_{\infty,1}, \ldots, r_{\infty,n})$ is the only n-tuple within our framework to satisfy the set of equations (12). For this reason we will say that a power series is *algebraic* [42] if it is one of the terms of a solution to a set of equations such as (12), where there is no restriction on the sign of the numerical coefficients. We will call a power series *context-free* if the coefficients in the defining equations are all positive.

In particular, $r_{\infty,1}$, which we will henceforth call *the power series generated by* the grammar G of (12) with initial symbol α_1, is the power series associated with G in the manner described at the outset of § 2.3. Its support is the language $L(G)$ generated by G, and the coefficient $\langle r_{\infty,1}, f \rangle$ of a string $f \in F(V_T)$ determines the ambiguity of f with respect to G, in the way described above.

Notice that if an algebraic power series is context-free, it is positive, but not necessarily conversely. That is, a power series may be a term of the solution to a set of equations and may have only positive coefficients, but may not be a term of the solution to any set of equations with only positive coefficients.[1])

[1]) For example, using notions which will be defined below in § 3.1, the Hadamard square $s \odot s$, for $s \in \lambda_0$, has only positive coefficients (and has the same support as s) but it is not, in general, generated by a set of equations with only positive coefficients.

2.4. As examples of the process described above, consider the two grammars (18) and (19):

(18) $S \to bSS;\ S \to a$

(19) $S \to SbS;\ S \to a.$

Each of these grammars has only a single non-terminal; hence the corresponding set of equations will in each case consist of a single equation. Corresponding to (18) we have (20), and corresponding to (19) we have (21).

(20) $S = a + bSS$

(21) $S = a + SbS.$

The equations (19) and (20) correspond to (12), above, with $n = 1$. Both (19) and (20) meet the condition (13).

Consider first the grammar (18) represented in the form (20). Proceeding in the manner of the preceding section, we regard (20) as defining a mapping ψ such that $\psi(r) = a + brr$, where r is a power series. We then (corresponding to (15)) form the infinite sequence $\varrho_0, \varrho_1, \varrho_2, \ldots$ as follows:

(22)
$$\begin{aligned}
&\varrho_0 = r_0 = 0\\
&\varrho_1 = r_1 = a + br_0r_0 = a + b00 = a\\
&\varrho_2 = r_2 = a + br_1r_1 = a + baa\\
&\varrho_3 = r_3 = a + br_2r_2 = a + b(a + baa)(a + baa)\\
&\qquad\qquad = a + baa + babaa + bbaaa + bbaabaa\\
&\varrho_4 = r_4 = a + br_3r_3\\
&\ldots\ \ldots\\
&\ldots\ \ldots
\end{aligned}$$

Clearly for each j, j' such that $j' > j > 0$, we have $r_j \equiv r_{j'} \pmod{\deg j}$. Consequently the limit r_∞ is well-defined. This power series is the solution to equation (20), and its support is the language generated by the *CF* grammar (18). Notice that the power series r_∞, in this case, is characteristic, and its support is the set of *well-formed formulas* of the "*implicational calculus*" with one variable in parenthesis-free (Polish) notation (with the symbol a playing the role of propositional variable, and b the role of the operator "conditional").

Consider now the grammar (19) represented in the form (21). We regard (21) as defining a mapping ψ such that $\psi(r) = a + rbr$, where r is a power series. We now form the infinite sequence $\varrho_0, \varrho_1, \varrho_2, \ldots$:

(23)
$$\begin{aligned}
&\varrho_0 = r_0 = 0\\
&\varrho_1 = r_1 = a + r_0 b r_0 = a + 0b0 = a\\
&\varrho_2 = r_2 = a + r_1 b r_1 = a + aba\\
&\varrho_3 = r_3 = a + r_2 b r_2 = a + (a + aba)b(a + aba)\\
&\qquad\qquad = a + aba + 2ababa + abababa\\
&\varrho_4 = r_4 = a + r_3 b r_3\\
&\qquad = a + aba + a(ab)^2a + 5(ab)^3a + 6(ab)^4a + 6(ab)^5a +\\
&\qquad\quad 4(ab)^6a + (ab)^7a\\
&\cdot\cdot\quad\cdot\cdot\cdot\\
&\cdot\cdot\quad\cdot\cdot\cdot
\end{aligned}$$

Again, for each j, j' such that $j' > j > 0$, we have $r_j = r_{j'}$, (mod deg j), and the limit r_∞ is defined as the power series

$$(24)\quad r_\infty = \sum_n \begin{bmatrix}2n\\ n\end{bmatrix} \frac{1}{n+1}\,(ab)^n a = a + aba + 2(ab)^2a + 5(ab)^3a + 14(ab)^4a + 42(ab)^5a + \ldots$$

where
$$\begin{bmatrix}2n\\ n\end{bmatrix} = \frac{2n \times 2n-1 \times \ldots \times n+1}{1 \times 2 \times \ldots \times n}$$

The power series r_∞ of (24) is the solution to the equation (21), and its support is the language generated by the grammar (19). It is not, in this case, a characteristic power series. Taking the symbol a again as a propositional variable and b as the sign for "conditional", the grammar (19) is the set of rules for generating the well-formed formulas of the implicational calculus with one variable in ordinary notation, but with the parentheses omitted. The structural descriptions generated by (19) in the manner described in section 1 (cf. (3)) are of course unambiguous, since brackets are preserved, but the terminal strings formed by debracketization are ambiguous, and the degree of ambiguity of each generated terminal string is exactly its coefficient in r_∞ — thus *ababa* can be interpreted in two ways, either as (*ab*(*aba*)) or ((*aba*)*ba*), etc. A more general case has been treated by Raney [38] by Lagrange's inversion formula.

In (20) and (21) all coefficients are positive and the solution is therefore a positive power series. Consider, however, the set of equations consisting of the single member

(25)
$$S = a - SbS.$$

In this case we have the sequence

$$
\begin{aligned}
(26)\qquad & \varrho_0 = r_0 = 0 \\
& \varrho_1 = r_1 = a - r_0 b r_0 = a - 0b0 = a \\
& \varrho_2 = r_2 = a - r_1 b r_1 = a - aba \\
& \varrho_3 = r_3 = a - r_2 b r_2 = a - (a - aba)b(a - aba) \\
& \qquad\qquad = a - aba + 2ababa - abababa \\
& \cdot \\
& \cdot \\
& \cdot
\end{aligned}
$$

In fact the coefficients in ϱ_i of (26) are exactly those of ϱ_i of (23) except for sign — the coefficient of f in ϱ_i of (26) is positive just in case f has an even number of b's.

The power series r_∞ which is the solution to (25) is not positive and it is consequently not context-free (though its support happens to be a context-free language, in this case, in fact, the language with (19) as one of its grammars). We can, however, regard r_∞ as the difference between two context-free power-series $r_\infty{}^+$ and $r_\infty{}^-$; and, correspondingly, we can regard its support as the set of strings that are not generated the same number of times by a pair of CF grammars G^+ and G^- which generate $r_\infty{}^+$ and $r_\infty{}^-$, respectively. Suppose we set $S = S^+ - S^-$, so that (25) becomes

$$
\begin{aligned}
(27)\qquad S^+ - S^- &= a - (S^+ - S^-)b(S^+ - S^-) \\
&= a - (S^+bS^+ - S^+bS^- - S^-bS^+ + S^-bS^-) \\
&= a + S^+bS^- + S^-bS^+ - (S^+bS^+ - S^-bS^-).
\end{aligned}
$$

Consider now the set of equations

$$
\begin{aligned}
(28)\ (\text{i})\qquad & S^+ = a + S^+bS^- + S^-bS^+ \\
(\text{ii})\qquad & S^- = S^+bS^+ + S^-bS^-.
\end{aligned}
$$

This is a set of positive equations with two variables S^+ and S^-, and it will have as solution the pair $(r_\infty{}^+, r_\infty{}^-)$, where $r_\infty{}^+$ is the limit of the sequence $r_0{}^+, r_1{}^+, \ldots$ and $r_\infty{}^-$ the limit of the sequence $r_0{}^-, r_1{}^-, \ldots$ of (29):

$$
\begin{aligned}
(29)\qquad & \varrho_0 = (r_0{}^+, r_0{}^-) = (0, 0) \\
& \varrho_1 = (r_1{}^+, r_1{}^-) = (a, 0) \\
& \varrho_2 = (r_2{}^+, r_2{}^-) = (a, aba).
\end{aligned}
$$

It is clear that where r_∞ is the solution to (25), $r_\infty = r_\infty{}^+ - r_\infty{}^-$. But, furthermore, $r_\infty{}^+$ is the power series generated by the CF grammar G^+ with the initial symbol S^+ and the grammar (28i); and $r_\infty{}^-$ is the power

series generated by the *CF* grammar G^- with the initial symbol S^- and the grammar (28ii).

In a similar manner, any algebraic power series can be represented (in infinitely many different ways) as the difference of two context-free power series, and its support can be regarded, therefore, as the set of strings which are not generated the same number of times by two *CF* grammars. This is as close as we can come to a concrete interpretation for the general notion of algebraic power series.

More generally, the same construction could be carried out for an arbitrary ring of coefficients instead of the ring of natural numbers used above. This is a still unexplored domain. For instance, if the coefficients are taken modulo a prime p (i.e., if we consider as "non-produced" the strings produced a multiple of p times), the formal power series $\Sigma_{n>0} z^{p^n}$ in the single terminal z is algebraic [27], although its support cannot be the support of any of the power series introduced above.

3. Further Operations on Formal Power Series

3.1. In § 2.2 we observed that the set of power series is closed under the operations of addition, product, and multiplication by an integer. We pointed out that the support of $r + r'$ is the union of the supports of r and r', and that the support of rr' is the set product of the supports of r and r', provided that the coefficients are non-negative. We will now turn to two other operations under which the set of power series is closed, and consider the corresponding set-theoretic interpretation for the supports.

It is standard terminology to say that r is *quasi-regular* if $\langle r, e\rangle = 0$. Then $r^{n'} \equiv 0 \pmod{\deg n}$ for $0 < n < n'$ and the element $r^* = \lim_{n\to\infty} \Sigma_{0<n'<n} r^{n'}$ is well-defined. Furthermore, r^* satisfies the identity

$$r + r^*r = r + rr^* = r^*, \tag{30}$$

which determines it uniquely. Thus r^* is usually called the *quasi-inverse* of r. This notion relates directly to the more familiar notion of an inverse by the remark that if $r' = e - r$ and $r'' = e + r^*$, then $r'r'' = (e - r)(e + r^*) = e - r + r^* - rr^* = e = r''r'$, that is, $r'' = r'^{-1}$. Conversely, given r' such that $\langle r', e\rangle = 1$, we can write it as $r' = e - r$, where r is quasi-regular, so that $e + r^*$ is the inverse of r''.

Note that by the very definition of r^*, this power series has only non-

negative coefficients if r does, and that Sup $r^* = (\text{Sup } r)^*$, where on the right side of the equation the star denotes Kleene's star operation [21].

In particular if V is an arbitrary set of letters and if the power series v is defined by $\langle v, x\rangle = 1$ if $x \in V = 0$ if $x \notin V$ (i.e., if v is the characteristic function of V), $e + V^*$ (in Kleene's sense) is the set of all words generated by the letters of V and $e + v^* = (e - v)^{-1}$ is the characteristic function of this set. This follows from the fact that any word $f \in V^*$ appears once and only once in the infinite sum $\Sigma_{n>0} V^n$. Consequently, when we know the characteristic function r of a set of strings, we are able to write also the characteristic function $(1 - V_T)^{-1} - r$ of its complement.

It is worth mentioning that in this case the latter has non-negative coefficients and although it is algebraic in the sense defined above, it is not necessarily context-free.

The second operation that we define is the *Hadamard product*, thus generalizing in one of the possible ways the usual notion of classical analysis. The definition that we give differs from the various extensions to the case of several variables that occur in the literature, but it seems to be most natural extension for non-commutative power series.

Where r and r' are two power series, their *Hadamard product* $r \odot r'$ will be the power series with coefficients

$$\langle r \odot r', f\rangle = \langle r, f\rangle \langle r', f\rangle \tag{31}$$

identically for all strings f. Hence Sup $(r \odot r') = (\text{Sup } r) \cap (\text{Sup } r')$, and $r \odot r'$ is a characteristic function if r and r' are.

Finally we introduce the following notation: given a string $x_{i_1} x_{i_2} \ldots, x_{i_{n-1}} x_{i_n} = f (x_{i_j} \in V)$ we define $\tilde{f}$ (the mirror image of f) to be the string

$$\tilde{f} = x_{i_n} x_{i_{n-1}} \cdots x_{i_2} x_{i_1} \tag{32}$$

Clearly $\tilde{\tilde{f}} = f$ and the relation $ff' = f''$ implies $\tilde{f}'' = \tilde{f}'\tilde{f}$. Formally this mapping is an *involutory anti-automorphism* of the ring of power series and it can be proved to be uniquely characterized by this property (up to a permutation of the elements of V).

3.2. The notation just introduced will be used later on for simplifying the description of grammars in the following way. Suppose that a grammar G contains the rules

$$\begin{aligned} \alpha_1 &= \pi_1 \alpha_2 \pi_2 + \pi_1 \pi_2 + \pi_3 \\ \alpha_2 &= \alpha_2 \pi_4 + \pi_4 \end{aligned} \tag{33}$$

where the π_j's are polynomial expressions in $V - \{\alpha_1\}$. Then the second rule implies

$$\alpha_2 = \left(\sum_{n>0} \pi_4{}^n\right) \tag{34}$$

and the rules (33) can be replaced by the simpler rule

$$\alpha_1 = \pi_1(1 - \pi_4)^{-1}\pi_2 + \pi_3. \tag{35}$$

We can, in fact, give a linguistic interpretation to this simplified form of description. Thus, for example, a pair of rules of the form $\alpha_1 \to f_1\alpha_2 f_2$, $\alpha_2 \to \alpha_2\alpha_2$ (that is, a pair which can now be given in the form: $\alpha_1 \to f_1(1 - \alpha_2)^{-1}f_2$) can be regarded as constituting, in effect, a rule schema: $\alpha_1 \to f_1\alpha_2{}^n f_2$ ($n = (1, 2, \ldots)$. With this reinterpretation, the grammar, though still finitely specified by rule schemata, consists of an infinite number of rules. But now recall the manner in which a structural description (a labelled bracketing) is assigned to a terminal string generated by a *CF* grammar (see above, § 1). A grammar specified by the rule schema given above can generate a structural description of the form

$$\text{--- } [_{\alpha_1} f_1 [_{\alpha_2} p_1][_{\alpha_2} p_2] \ldots [_{\alpha_2} p_n] f_2] \text{ ---} \tag{36}$$

for each n, where each p_k is derived from α_2. In the sentence (terminal string) with this structural description, each $\bar{p}_k$ is a phrase of type α_2, where $\bar{p}_k$ is formed by debracketization of p_k. The successive phrases $\bar{p}_1, \ldots, \bar{p}_n$ form a "coordinate construction", which, taken together with the strings formed ultimately from f_1 and f_2, is a construction of the type α_1. This is the natural way to extend *CF* grammars to accommodate true coordination, as, e.g., where a string of adjectives of arbitrary length may appear in predicate position with no internal structure defined on them. Cf. Chomsky [10].

3.3. Let us try to relate what we have done so far to classical analysis, writing $\varphi f = \varphi f'$ for any two strings f and f' if they contain exactly the same number of each of the letters (terminal or not).

Clearly φ extends to a mapping of our non-commutative power series onto the ring of the ordinary (commutative) *formal* power series with integral coefficients, and it is easily seen that φ is a homomorphism. For example, if $\alpha = a + b\alpha\alpha$, we have $\varphi\alpha = \varphi a + \varphi b\varphi\alpha\varphi\alpha$, and $\varphi\alpha$ is the ordinary power series

$$\varphi\alpha = (\varphi a)^{n+1}(\varphi b)^n \begin{bmatrix} 2n \\ n \end{bmatrix} \frac{1}{n+1} \tag{37}$$

in the ordinary variables φa, φb. (Here if $\alpha' = a + \alpha' b \alpha'$, we would also have $\varphi\alpha' = \varphi\alpha$).

Furthermore, it can be shown directly from the way our power series are obtained that the coefficients do not grow faster than an exponential function of the degree (length) of the strings. Thus the image φ of any one of our power-series is in fact an ordinary convergent Taylor series expansion of an algebraic function.

Reciprocally, if we are given (ordinary) variables $\bar{x}_1, \ldots, \bar{x}_n$, an (ordinary) algebraic function of this quantity $\bar{y}$ is defined by a polynomial in $\bar{y}$ and the $\bar{x}_i$; and in case $\bar{y}$ admits a development in Taylor series (with integral coefficients) around zero in the $\bar{x}_i$'s, we can associate with it infinitely many formal power series β such that $\varphi\beta = \bar{y}$ and β is defined by formal equations. For instance: starting from the algebraic function $\bar{y}$ of $\bar{a}$ and $\bar{b}$ defined by $\bar{y}^2\bar{b} - \bar{y} + a = 0$, we obtain the two examples given above, and also formal power series

$$\alpha = a + b\alpha\alpha + \pi\alpha - \alpha\pi \tag{38}$$

where π is an arbitrary polynomial in a and b. Thus, e.g., take $\pi = b$. Then

$$\begin{aligned} &\alpha_0 = a \\ &\alpha_1 = a + baa + ba - ab \\ &\cdots\cdots \\ &\text{etc.} \end{aligned} \tag{39}$$

3.4. Let us conclude by indicating some connections between our considerations and Lyndon's theory of equations in a free group (Lyndon, 1960). Let $\{x_i\}$ $(1 \leq i \leq n)$ be a terminal vocabulary, ξ a non-terminal letter and let w be a product of terms of the form $1 - x_i$, $(1 - x_i)^{-1}$, $1 - \xi$, $(1 - \xi)^{-1}$. We define $\deg(w) = d_+ - d_-$ where d_+ and d_- are the number of factors $1 - \xi$ and $(1 - \xi)^{-1}$ in w. Thus, for instance, for $w = (1 - x_2)(1 - x_1)(1 - \xi)(1 - x_i)^{-1}(1 - \xi_i)^{-1}(1 - x_2)^{-1}$, one has $\deg(w) = 1 - 1 = 0$.

As is well-known, the elements $1 - x_i$ generate (by multiplication) a free group G. The relation $w = 1$ may be considered as an equation in the unknown ξ. In our terminology a *solution* of $w = 1$ would be a power series ξ_0 in the x_i's such that $w = 1$ identically when ξ_0 is substituted for ξ in w; ξ_0 will be a *group solution* if, furthermore, $1 - \xi_0 \in G$; i.e., if $1 - \xi_0$ is itself expressible as a product of terms $(i - x_i)^{\pm 1}$. R.C. Lyndon

has proven the very remarkable result that the *totality* of the *group solutions* can be obtained algorithmically.

Let us relate part of this question to our remarks in § 2.3. For this we introduce the new symbols $\xi_i (1 \leq i \leq n)$, η, and equations

$$\xi_i = x_i + \xi_i x_i; \quad y = \xi^2 + \xi\eta \tag{1}$$

so that $(1 - x_i)^{-1} = 1 + \xi_i$ and $(1 - \xi)^{-1} = 1 + \xi + \xi^2 + \xi\eta$

Substituting these expressions in $w = 1$ and simplifying, we obtain a relation

$$(\deg (w))\, \xi = p' \tag{2}$$

where p' is a polynomial in the variables x_i, ξ_i, η having no term of degree less than 2.

Hence if $\deg (w) \neq 0$ the system (1), (2) has one and only one solution in power series (the fact that the coefficients are eventually rational instead of integral numbers is irrelevant to the proof in § 2.3) and since the group solutions are a subset of the power series solutions, we have verified directly that if $\deg w \neq 0$, the free group equation $w = 1$ has at most one solution.

On the contrary, if $\deg w = 0$ (as for instance for the equation $w = (1 - \xi)(1 - x_i)(1 - \xi)^{-1}(1 - x_i)^{-1} = 1$) our approach entirely collapses and says nothing even about the unrestricted solutions of $w = 1$.

For instance $(1 - x_i)(1 - \xi)(1 - x_i)^{\varepsilon}(1 - \xi)^{-1} = 1$ has no solution if $\varepsilon \neq -1$ and has an infinity of *group* solutions if $\varepsilon = -1$, viz. $1 - \xi = (1 - x_i)^{\pm n}$ $(n > 0)$. Indeed, then, the equation can equivalently be written $\xi x_1 = x_1 \xi$ (which has as solutions, in our sense, all the power series in x_1).

Of course, the case $\deg w = 0$ is precisely that in which, the unknown $1 - \xi$ disappears when taking the commutative image as in § 3.3 and it is the non-trivial case from a group theoretic point of view.

4. Types of *CF* Grammars and Their Generative Properties

4.1. In terms of conditions on the rules that constitute them, we can define several categories of *CF* grammars that are of particular interest. In the following we will use $\alpha, \beta, \ldots$ for non-terminal symbols; $f, g, \ldots$ for terminal strings (possibly null); and φ, ψ for arbitrary strings. Recall that we have

excluded the possibility of rules of the form $\alpha \to e$ or $\alpha \to \beta$, remarking that this restriction does not affect generative capacity. We will describe *CF* grammars in terms of rules or equations, whichever is more convenient.

If the grammar G contains no non-terminal α from which it is possible to derive both a string f' and a string $f\alpha g$, then the terminal language $L(G)$ generated by G will be finite. In this case, G will be called a *polynomial grammar*.

Consider now grammatical rules of the following kinds:

(40) (i) $\alpha \to f\beta$ (right-linear)
(ii) $\alpha \to \beta f$ (left-linear)
(iii) $\alpha \to f\beta g$ (linear)
(iv) $\alpha \to f$ (terminating)

A grammar containing only right-linear and terminating rules or only left-linear and terminating rules will be called a *one-sided linear* grammar.

A grammar containing only rules of the type (40) will be called *linear*. Suppose that G contains only rules of the type (40) and of the type $\alpha_1 \to \varphi$, where α_1 is the initial symbol of G; and that, furthermore, it contains no rule $\beta \to \varphi\alpha_1\psi$. Thus the defining equation for α_1 is $\alpha_1 = \pi_1$, where π_1 is a polynomial not involving α_1. Such a grammar will be called *meta-linear*.

Given a grammar G (i.e., a set of positive equations) which is polynomial, one-sided linear, linear, meta-linear or context-free, we will say that the power series r which is the principle term of its solution (i.e., which it *generates*, in the sense defined in § 2.3) and the language Sup r which it generates are, respectively, polynomial, one-sided linear, linear, meta-linear or context-free. These families of power-series will be designated, respectively, $\mathscr{P}^+$, $\mathscr{L}_0^+$, $\mathscr{L}^+$, $\mathscr{L}_m^+$, $\mathscr{I}^+$; and for each family $\mathscr{F}$ the family of supports of $\mathscr{F}$ will be designated Sup $(\mathscr{F})$.

Notice that Sup $(\mathscr{P}^+)$ is just the family of finite sets, and that Sup $(\mathscr{L}_0^+)$ is the family of regular events, in the sense of Kleene [21] (cf. Chomsky, [7] — note that the class of regular events is closed under reflection).

We consider now certain elementary properties of these families of languages.

It is, first of all, immediate that the following inclusion relations hold among these families:

(41) $\mathrm{Sup}(\mathscr{P}^+) \subset \mathrm{Sup}(\mathscr{L}_0^+) < \mathrm{Sup}(\mathscr{L}^+) < \mathrm{Sup}(\mathscr{L}_m^+) < \mathrm{Sup}(\mathscr{I}^+)$.

Furthermore, in each of these cases inclusion can be strengthened to proper inclusion. Thus we have:

PROPERTY 1.

$$\text{Sup}(\mathscr{P}^+) \subsetneq \text{Sup}(\mathscr{L}_0{}^+) \subsetneq \text{Sup}(\mathscr{L}^+) \subsetneq \text{Sup}(\mathscr{L}_m{}^+) \subsetneq \text{Sup}(\mathscr{I}^+).$$

The simplest example of a language in $\text{Sup}(\mathscr{L}^+)$ but not in $\text{Sup}(\mathscr{L}_0{}^+)$ is the set of all strings $\{a^n b a^n\}$ $(a, b \in V_T)$. This is generated by the grammar: $\alpha = a\alpha a + b$, and is easily shown not to be a regular event. The product of languages in $\text{Sup}(\mathscr{L}^+)$ is always in $\text{Sup}(\mathscr{L}_m{}^+)$, but not in general in $\text{Sup}(\mathscr{L}^+)$. The language L_{IC} of our example (18) above with the grammar:

$$\alpha = a + b\alpha\alpha \tag{42}$$

and consisting of the set of well-formed formulas of the implicational calculus with one free variable in Polish notation is in $\text{Sup}(\mathscr{I}^+)$ but not in $\text{Sup}(\mathscr{L}_m{}^+)$. This follows from the fact that L_{IC} contains all strings of the form

$$b^{m_1} a^{m_1} b^{m_2} a^{m_2} \dots b^{m_k} a^{m_k} a, \tag{43}$$

for each $k \geq 1$, $m_i \geq 1$. But each string in L_{IC} contains n occurrences of b and $n + 1$ occurrences of a, for some $n \geq 1$. Consequently, for a fixed integer k, to generate all strings of the form (43), it must be possible to derive from the initial symbol of the grammar of L_{IC} a string φ containing k occurrences of non-terminals. Consequently, this grammar cannot be metalinear.

For empirical interpretation of the theory of *CF* grammars, the relation between $\text{Sup}(\mathscr{I}^+)$ and $\text{Sup}(\mathscr{L}_0{}^+)$ is of particular importance, since a finite device incorporating the instructions of a *CF* grammar G generating $L(G)$ as a representation of its intrinsic competence, will be able to interpret only the sentences of some fixedsubset $R \in \text{Sup}\,(\mathscr{L}_0{}^+)$ of $L(G) \in \text{Sup}(\mathscr{I}^+)$ (with fixed supplementary aids). This relation can be described precisely in terms of certain formal features of the structural descriptions (labelled bracketings) generated by *CF* grammars — cf. § 1. Let us say that G is a *self-embedding grammar* if it generates a structural description of the form

$$\dots [_\alpha \varphi [_\alpha \psi] \chi] \dots, \tag{44}$$

where φ and χ contain non-null terminals, and where ψ is a properly bracketed expression. Then we have the following result:

THEOREM 1a.
$L \notin \mathscr{L}_0^+$ if and only if every CF grammar that generates L is self-embedding.
Chomsky [9]. This result can be extended in the following way. Define the *degree of self-embedding* of a structural description D as the largest N such that D contains a subconfiguration:
$[_\alpha\varphi_1[_\alpha\varphi_2[_\alpha \ldots [_\alpha\varphi_{N+1}]\varphi_{N+2}] \ldots]\varphi_{2N+1}]$ where each φ_i contains non-null terminals. Then there is a one-one effective mapping Φ of $\{(G,n) : G$ a *CF* grammar, $n \geq 1\}$ into the set of one-sided linear grammars and a one-one effective mapping Ψ of the set Δ of structural descriptions into Δ such that:

THEOREM 1b.
For each $L \in Sup(\mathscr{S}^+)$, there is a CF grammar G generating L such that for each N, Φ (G,N) generates f with the structural description D if and only if G generates the terminal string f with the structural description $\Psi(D)$, where $\Psi(D)$ has degree of self-embedding $\leq N$.
Chomsky [8]. Thus, intuitively, we can, given G, construct a finite device $\Phi(G,N)$ that will recognize the structure of a string f generated by G just insofar as the degree of self-embedding of a particular structural description of f does not exceed N. This fact suggests certain empirical consequences. For discussion, cf. Chomsky [10], Miller and Chomsky [29].

4.2. We consider now various closure properties of these families of languages.

The families of power series defined above can be given the following algebraic characterization. $\mathscr{P}^+$ is a *semi-ring.*[1]) $\mathscr{L}_0^+$ is the smallest semi-ring containing $\mathscr{P}^+$ and closed by quasi-inversion of quasi-regular elements. $\mathscr{L}^+$ is a *module*, and $\mathscr{L}_m^+$ is the smallest semi-ring containing it. The full set $\mathscr{S}^+$ is a semi-ring closed by quasi-inversion of quasi-regular elements.

Correspondingly, we have the following properties of the supports: $Sup(\mathscr{P})$ is closed under set union and set product; $Sup(\mathscr{L}_0^+)$ is the smallest set containing the finite sets and closed under the operations of set union, set product, and the star operation described in § 3.1 [21]; $Sup(\mathscr{L}^+)$ is closed under set union, but not set product; $Sup(\mathscr{L}_m^+)$ is the smallest set

[1]) The notion of semi-ring generalizes to that of ring in that the additive structure is only a monoid (not necessarily group) structure. A typical semi-ring is the so-called "Boolean ring" with two elements 0 and 1 and the rules
$(0 = 0 + 0 = 00 = 01 = 10;\ 1 = 0 + 1 = 1 + 0 = 1 + 1 = 11)$.

containing the sets of Sup($\mathscr{L}^+$) and closed under set product as well (this is, of course, the motivation behind the construction of $\mathscr{L}_m{}^+$); The full set Sup($\mathscr{I}^+$) is closed by union, product and the star operation.

These properties are immediate, and it is natural to inquire into closure under the other elementary operations on sets, namely, intersection and complementation. It is obvious that Sup($\mathscr{P}^+$) is closed under intersection, and it is well-known that the class Sup($\mathscr{L}_0{}^+$) of regular events is closed under intersection and complementation.

For the other families, we have the following results. The family Sup($\mathscr{I}^+$) of all *CF* languages is not closed under intersection and hence (since it is closed under union) not closed under complementation [40], [2]. The example given, in each of these references, consists of a pair of meta-linear languages whose intersection is not context-free. Hence it follows that Sup ($\mathscr{L}_m{}^+$) is also not closed under intersection or, consequently, complementation. This result can be strengthened to cover linear grammars, in fact (for intersection) even linear grammars with a single non-terminal.

To see this, consider the grammars G_1 and G_2 defined as in (45) and (46) respectively:

(45) $$\alpha = aa\alpha c + b\alpha c + bc$$

(46) $$\alpha = a\alpha cc + a\alpha b + ab.$$

G_1 and G_2 are each linear with a single non-terminal. But the intersection of the languages that they generate is the set of strings $[a^{2n}b^n a^{2n}]$, which is not context-free. This example (along with the fact that these families are closed under union) establishes that

PROPERTY 2.

The families Sup($\mathscr{L}^+$), Sup($\mathscr{L}_m{}^+$), Sup($\mathscr{I}^+$) are not closed under either intersection or complementation; the intersection of two sets in one of these families may not even be in Sup($\mathscr{I}^+$), even when the sets in question are generated by grammars with a single non-terminal.

Presumably the complement of a language of Sup($\mathscr{L}^+$) or of Sup($\mathscr{L}_m{}^+$) is not context-free (i.e., is not a member of Sup($\mathscr{I}^+$)). However, we have no examples to show this.

Thus of the classes of languages discussed above, only the regular events (and the finite sets) are closed under formation of intersections. However, the intersection of a regular event and a context-free grammar

is again a context-free language [2]. We have in fact, the following stronger result which extends a well-known theorem of classical analysis due to R. Jungen [20].

THEOREM 2.
Suppose that $r_1 \in \lambda_0^+$. *Let* U^+ *be one of the families* $\mathscr{P}^+, \lambda_0^+, \lambda^+, \lambda_m^+, \mathscr{S}^+$. *Let* $r_1 \odot r_2$ *be the Hadamard product of* r_1, r_2 (*cf.* § 3.1). *Then* $r_1 \odot r_2 \in U^+$, *for every* $r_2 \in U^+$. *Furthermore, if* $r_2, r_3 \in \lambda_0^+$, *then* $r_2 \odot r_3 \in \lambda_0^+$.

Cf. Schützenberger [46]. It follows that the intersection of a language of $\mathrm{Sup}(U^+)$ with a regular event is in $\mathrm{Sup}(U^+)$, for each U^+. The proof of this result, which is related to a similar result concerning closure under transduction, will be outlined in § 8, below.

4.3. The category of linear grammars is of particular interest, as we will see directly, and we will now make a few preliminary observations concerning it. Notice that if L is a language generated by a linear grammar, we can find a vocabulary V' disjoint from V_T, two homomorphic mappings α, α' of $F(V')$ into $F(V_T)$, a regular event R in V', and a finite set $C \subset F(V_T)$ such that L consists of exactly the strings $f = \alpha(g)c\,\alpha'(\tilde{g})$, where $g \in R$, $\tilde{g}$ is the reflection of g, and $c \in C$. Thus a finite process dealing with a collection of pairs of strings or a pair of coordinated finite processes can, in general, be correlated to a linear grammar and studied in this way.

Equivalently, we can characterize a linear language in the following, slightly different way. Let $V' = V^+ \cup V^-$ ($V^+ = \{v_i : 0 \leq i \leq n\}$; $V^- = \{v_i : -n \leq i \leq -1\}$. Where $f \in F(V^+)$, let us define $\tilde{f}$ as the result of substituting v_{-i} for v_i in f, throughout. Then a linear language L is determined by choice of a homomorphic mapping β of $F(V')$ into $F(V_T)$, a regular event R in V^+, and a finite set $C \subset F(V_T)$. L is now the set of strings $\beta(f)c\beta(\tilde{f})$, where $f \in R$ and $c \in C$. We will use this alternative characterization below.

We can now determine special classes of linear languages by imposing further conditions on the underlying regular event R, the mappings α, α', and the class C. In particular, in applications below we will be concerned with the case in which R is simply a free monoid (a regular event defined by a single-state automaton) and where C contains just $c \in V_T$, where $\alpha(f) \neq \varphi c \psi \neq \alpha'(f)$. We will call grammars defined by this condition *minimal linear grammars.*

A minimal linear grammar contains a single non-terminal symbol S and a single terminating rule $S \to c$, and no non-terminating rule $S \to \varphi c \psi$. Thus each string of the language it generates has the designated "central marker" c. This is the simplest set of languages in our framework beyond the regular events, and we will see that they differ markedly from regular events in many formal properties.

For later reference, we give now one particular result concerning minimal linear grammars. Let us take V', $V_T = W \cup \{c\}$ $(c \notin W)$, α and α' as above. Let G be the minimal linear grammar defined by α, α' and generating $L(G)$. Thus G has the defining equation

$$\beta = c + \Sigma\{\alpha(v)\beta\alpha'(v) : v \in V'\} \tag{47}$$

where α, α' are mappings of $F(V')$ into $F(W)$. Then we have:

THEOREM 3.
If α is a monomorphism (isomorphism into), then the complement $F(V_T) \setminus L(G)$ of $L(G)$ with respect to $F(V_T)$ is generated by an unambiguous linear grammar.

Proof: Let $A = \alpha(V')$, $F(A) = \alpha F(V')$, and for any set $F \subset F(W)$, let $F^+ = \{f \in F : f \neq e\}$.

Clearly there is a partition: $F(V_T) \setminus L(G) = L \cup L'$, such that

(48)

$$L = fcf' : f \in F^+(A), f' \in F(W), fcf' \notin L(G);$$
$$L' = F(W) \cup cF(W) \cup ((F(W) \setminus F(A))\, cF(W) \cup F(V_T)cF(V_T)cF(V_T).$$

But L' is a regular event. Hence it suffices to show that L is generated by an unambiguous linear grammar.

Since α is a monomorphism, there exists an isomorphism $\bar{\alpha} : F(A) \to F(V')$. We extend α' to $F^+(A)$ by defining $\alpha' a = \alpha'(\bar{\alpha} a)$, for $a \in F^+(A)$.

Suppose that $acf' \in L$. Thus $\in F^+(A)$, $f' \in F(W)$, and $f' \neq \alpha' a$. By definition there are just three mutually exclusive possibilities for acf'.

(49) (i) $f' \in F^+(W)\alpha' a$
(ii) $\alpha' a \in F^+(W) f'$
(iii) $a = a_1 a_2 a_3$ and $f' = hwg\alpha' a_1$ (where $a_1, a_3 \in F(A)$; $a_2 \varepsilon A$; $w \in W$; $h,g \in F(W)$; $\alpha' a_2 \in F^+(W)g$; $\alpha' a_2 \in F(W)wg$).

(49i) is the case in which f' has $\alpha' a$ as a proper right factor;
(49ii) is the case in which $\alpha' a$ has f' as a proper right factor;

(49iii) is the case in which $\alpha' a$ and f' have as their common maximal right factor the string $g\alpha' a_1$, which is a proper substring of both $\alpha' a$ and f'. Thus the three cases are mutually exclusive and exhaustive, and we have a partitioning of L into the three subsets L_1, L_2, L_3, consisting of the strings meeting (i)–(iii), respectively. What we now have to show is that each of L_1, L_2, L_3 is generated by an unambiguous linear grammar.

In the case of L_1 and L_2 this fact is obvious. Let $\bar{A} = \Sigma\{a : a \in A\}$ and $\bar{W} = \Sigma\{w : w \in W\}$. Then L_1 is generated by the grammar (50) and L_2 by the grammar (51) (cf. § 3.2).

$$\beta = \Sigma\{a\beta\alpha' a : a \in A\} + c(1 - \bar{W})^{-1} \tag{50}$$

$$\beta = \Sigma\{a\beta\alpha' a : a \in A\} + (1 - \bar{A})^{-1}c \tag{51}$$

Consider now the case of L_3. For each $a \in A$, let us denote by $B(a)$ the set of all strings wg ($w \in W$, $g \in F(W)$) such that $\alpha' a \in F^+(W)g$ and $\alpha' a \notin F(W)wg$. Clearly $B(a)$ is always a finite set, since g is shorter than $\alpha' a$. We can now generate L_3 by the unambiguous linear grammar with the equations:

$$\begin{aligned} \beta_1 &= \Sigma\{a\beta_1\alpha' a : a \in A\} + \Sigma\{a\beta_2 b : a \in A, b \in B(a)\} \\ \beta_2 &= c + c(1 - \bar{W})^{-1} + (1 - \bar{A})^{-1}c + \Sigma\{a\beta_2 w : a \in A, w \in W\}. \end{aligned} \tag{52}$$

Verification is straightforward. But now we have given $F(V_T) \setminus L(G)$ as the union of the four disjoint sets L_1, L_2, L_3, L', each of which has an unambiguous linear grammar. Consequently, $F(V_T) \setminus L(G)$ itself has an unambiguous linear grammar, as was to be proven.

Notice that if we had taken α originally as an "information-lossless transduction" [43] instead of as a monomorphism, we could prove a result differing from *Theorem 3* only in that the linear grammar constructed would have bounded ambiguity, rather than no ambiguity.

4.4. We have considered several subfamilies of the class of *CF* grammars, classifying on the basis of structural properties of the defining rules. There are other principles of classification that might be considered. Thus, for example, it might be worthwhile to isolate the class of the *star grammars* (*languages*) characterized as follows: G is a star grammar if associated with each non-terminal α_i of G there is a set Σ_i of non-terminals and three terminal strings f_i, f'_i, f''_i, and G contains all and only the rules: $\alpha_i \to f''_i$, $\alpha_i \to f_i\alpha_j f'_i$ ($\alpha_j \in \Sigma_i$), $\alpha_j \to \alpha_k\alpha_l$ ($\alpha_j, \alpha_k, \alpha_l \in \Sigma_i$). These are, in a sense, the most "structureless" *CF* grammars. The interest of

these languages lies in the fact that the equations defining the associated power series are expressible using in an essential manner only the quasi-inverse and addition, as we have observed in § 3.2. Notice, in particular, that the non-metalinear language L_{IC} defined by (42) is a star language. We have suggested a linguistic interpretation for the notion "star language" in § 3.2.

Another principle of classification might be in terms of the number of non-terminals in the minimal defining grammar of a certain power series. However, it does not seem likely that interesting properties of language can correlate with a measure so insensitive to structural features of grammars as this (except for the special case of the languages defined by grammars with only *one* non-terminal), because for monoids, as distinct from groups, the gross numerical parameters do not relate in an interesting way to the fine structure. Notice, incidentally, that for any finite N we can construct a regular event which cannot be generated by a *CF* grammar with less then N non-terminal symbols.

Another principle of classification is suggested by consideration of dependencies among subparts of the grammar. Let us call a *CF* grammar *irreducible* if no proper subset of the set of defining equations constitutes a *CF* grammar (recall that terminal strings must be derivable from each non-initial non-terminal of a *CF* grammar); otherwise, *reducible.* If a *CF* grammar is reducible, in this sense, there must be proper subsets Σ_1 of its rules and Σ_2 of its non-terminals, such that only rules of Σ_1 are involved in extending derivations to terminated derivations at points where symbols of Σ_2 appear in lines of derivations.

One particular extreme form of reducibility has been studied by Ginsburg and Rice (18). Following them, let us call a *CF* grammar G *sequential* if its non-terminals can be ordered as $\alpha_1, \ldots, \alpha_n$ (where α_1 is the initial symbol) in such a way that there is no rule $\alpha_i \to \varphi\alpha_j\psi$ for $j < i$. The solution to a sequential grammar is particularly easy to determine by the iterative procedure described in § 2.3 by successive elimination of variables.

Concerning the family $\mathscr{S}^+$ of sequential grammars and the family $\mathrm{Sup}(\mathscr{S}^+)$ of their supports, Ginsburg and Rice establish the following results, paralleling those mentioned above. First, it is clear that $\mathscr{S}^+$, like $\mathscr{I}^+$ is a semi-ring closed by quasi-inversion of quasi-regular elements. Correspondingly, $\mathrm{Sup}(\mathscr{S}^+)$ is closed by union, product, and the star operation. From this fact, and the fact that $\mathscr{P}^+ \subset \mathscr{S}^+$, it follows that $\mathrm{Sup}(\mathscr{L}_0^+) \subset \mathrm{Sup}(\mathscr{S}^+)$. Furthermore, the inclusion is proper, as we can

see from the grammar (42), which, since it contains a single terminal, is sequential. In fact, we have

$$\text{Sup}(\mathscr{L}_0^+) \subsetneq \text{Sup}(\mathscr{S}^+) \subsetneq \text{Sup}(\mathscr{I}^+). \tag{53}$$

Ginsburg and Rice show that there is no sequential grammar for the language with the vocabulary $\{a, b, c, d\}$ and containing the string

$$a^{n_{2k-1}} d \dots db^{n_2} da^{n_1} ca^{n_1} db^{n_2} d \dots b^{n_{2k-2}} da^{n_{2k-1}} \tag{54}$$

(which is symmetrical about c) for each sequence $(k, n_1, \dots, n_{2k-1})$ of positive integers, although this language is generated by the grammar.

$$\begin{aligned} \alpha &= ad\beta da + a\alpha a + aca \\ \beta &= b\beta b + bd\alpha db. \end{aligned} \tag{55}$$

There is no stronger relation than (53) between $\text{Sup}(\mathscr{S}^+)$ and the families of *Property* 1, § 4.1, however. The grammar (55) is in fact linear, though not sequential, so that $\text{Sup}(\mathscr{L}^+) \not\subset \text{Sup}(\mathscr{S}^+)$; and the grammar (42) is sequential but not meta-linear, so that $\text{Sup}(\mathscr{S}^+) \not\subset \text{Sup}(\mathscr{L}_m^+)$.

Since the grammars (45) and (46) are sequential, we see that *Property 2* (but not *Theorem 2*) can be extended to $\text{Sup}(\mathscr{S}^+)$. For further results on sequential languages, see Ginsburg and Rose [19], Shamir [51].

5. An Alternative Characterization of Families of *CF* Languages

In this section we will present a rather different approach to the definition of families of languages, and we will show how it interrelates with the classification presented above. We rely here on the two fundamental notions: *standard regular event* and *Dyck language*, which we now define.

A *standard regular event* A is given by a finite alphabet X, two subsets J_1 and J_2 of (X, X), and the rule that $f \in A$ if and only if

$$\begin{aligned} &\text{(i)} && f \in xF(X) \cap F(X)x', \text{ where } (x, x') \in J_1 \\ &\text{(ii)} && f \notin F(X)xx'F(X), \text{ where } (x, x') \in J_2. \end{aligned} \tag{56}$$

Thus A is the set of all strings that begin and end with prescribed letters, and that contain no pair of consecutive letters belonging to J_2. It is, more technically, the intersection of the quasi-ideal determined by J_1 with the complement of the two-sided ideal generated by all products

xx' ($(x, x') \in J_2$). A is, in particular, what is sometimes called a "1-definite event" [21], [35].

We define the *Dyck language* D_{2n} on the $2n$ letters $x_{\pm i}$ ($1 \leq i \leq n$) as the set of all strings f which can be reduced to the empty string by repeated cancellation of consecutive pairs of letters $x_j x_{-j}$ ($-n \leq j \leq n$). The Dyck language is a very familiar mathematical object: if φ is the homomorphism of the free monoid generated by $\{x_{\pm i}\}$ onto the free group generated by the subset $\{x_i : i > 0\}$ that satisfies identically $(\varphi x_i)^{-1} = \varphi x_{-i}$, then D_{2n} is the kernel of φ, that is, the set of strings f such that $\varphi f = 1$.

Concerning these notions, we have the following results.

PROPOSITION 1.
For any regular event $B \subset F(Z)$, we can find a standard regular event A and a homomorphism $\alpha : F(X) \to F(Z)$, such that $B = \alpha A$.

It is worth mentioning that this representation can be chosen in such a way that not only $B = \alpha A$, but, furthermore, each string $f \in A$ has the same degree of ambiguity as the corresponding string $\alpha f \in B$. That is, if $B = \mathrm{Sup}(\beta)$, we can find γ such that $A = \mathrm{Sup}(\gamma)$ and for each f, $\langle \gamma, f \rangle = \langle \beta, \alpha f \rangle$.

We can generalize *Proposition 1* to *CF* languages, making use of the following property of D_{2n}.

PROPERTY 1. *D_{2n} is generated by an unambiguous CF grammar.*
To obtain an unambiguous grammar of D_{2n}, we introduce $2n + 1$ nonterminals $\alpha_{\pm i}$ ($1 \leq i \leq n$) and β. Consider now the $2n + 1$ equations

$$\text{(57) (i)} \qquad \alpha_i = x_i \Big(1 - \sum_{j \neq -i} \alpha_j\Big)^{-1} x_{-i}$$

$$\text{(ii)} \qquad \beta = (1 - \Sigma \alpha_i)^{-1}.$$

(Cf. § 3.2, for notation).

Intuitively, β can be interpreted as the sum of all strings that can be reduced to the empty string by successive cancellation of two consecutive letters $x_i x_{-i}$. Each α_i is the sum of all words in $\mathrm{Sup}(\beta)$ that begin by x_i and have no proper left (or right) factor in $\mathrm{Sup}(\beta)$. The equation (57i) implies that each $f \in \mathrm{Sup}(\alpha_i)$ has one and only one factorization

$$f = x_i f_1 f_2 \ldots f_m x_{-i} \tag{58}$$

where each f_j belongs to a well-defined set $\mathrm{Sup}(\alpha_j)$ (where j is not $-i$ because we want the initial letter x_i to cancel only with the final letter x_{-i}).

Similarly, each $f \in \mathrm{Sup}(\beta)$ has one and only one factorization $f = f_1 \ldots f_m$, where the f_j's belong to $\cup_i \mathrm{Sup}(\alpha_i)$.

We now have the following result, analogous to *Proposition 1.*

PROPOSITION 2.

Any CF language $L \subset F(Z)$ is given by an integer n, a standard regular event A on $X_{2n} = \{x_{\pm i} : 1 \leq i \leq n\}$, a homomorphism $\varphi : F(X_{2n}) \to F(Z)$, and the rule $L = \varphi(A \cap D_{2n})$.

[48], [49], [11], [12].

Again, as above, this statement implies that the strings are produced with the appropriate ambiguity. Furthermore, it is possible to choose J_1 such that $(x,x') \in J_1$ if x belongs to a certain subset of X (cf.[48]).

Special subfamilies of languages such as those considered above can be defined by imposition of conditions on the underlying standard regular event A and the homomorphism φ. Thus suppose that we take the standard regular event A on the alphabet $X \cup Y$ (where $X = \{x_{\pm i} : 1 \leq i \leq n\}$, $Y = \{y_{\pm i} : 1 \leq i \leq m\}$ defined by the following conditions on J_1 and J_2:

$$(59)\quad J_1 = \{(x_i, x_j) : i > 0\}$$
$$J_2 = \{(x_i, x_j) : \mathrm{sign}\,(i) \neq \mathrm{sign}\,(j)\} \cup \{(y_i, y_j) : i < 0 \text{ or } j > 0\} \cup \{(x_i, y_j) : i < 0 \text{ or } j < 0\} \cup \{(y_i, x_j) : i > 0 \text{ or } j > 0\}.$$

Thus every string has the form $fgg'f'$, where $f, f' \in F(X)$; $g, g' \in F(Y)$; f, g (respectively, f', g') contain only letters with positive (respectively, negative) indices. If we designate by X^+ and X^- the subsets of X consisting of letters with positive indices and negative indices, respectively (similarly, Y^+ and Y^-), we can describe the permitted and excluded transitions by the matrix (60), where the entry 1 (0) indicates that transition is (is not) permitted from the element labelling to row to that labelling the column, and where U is the matrix with all one's and 0 the matrix with all zeroes.

(60)

	Y^+	Y^-	X^+	X^-
Y^+	0	U	0	0
Y^-	0	0	0	U
X^+	U	0	U	0
X^-	0	0	0	U

But consider now the set $A \cap D_{XY}$ (where D_{XY} is the Dyck language

on the alphabet $X \cup Y$). If $fg \in A$ (where $f \in F(X^+ \cup Y^+)$, $g \in F(X^- \cup Y^-)$) meets the additional condition that $fg \in D_{XY}$, then g must be the mirror-image of f (up to a change of the sign of indices). That is, in the notation of the second paragraph of § 4.3, it must be the case that $g = \bar{f}$. Clearly, if α is a homomorphic mapping of $F(X \cup Y)$ into $F(V_T)$, then $\alpha(A \cap D_{XY})$ is a linear language. Furthermore, if we add the further condition that $y_i = e$ for $i < 0$ and $y_i = c$ for each $i > 0$, where $\alpha x_i \notin F(V_T)cF(V_T)$ for any i, then $L = \alpha(A \cap D_{XY})$ is a minimal linear language with c as designated center symbol, and every minimal linear language is given by such a choice of α. This gives an independent characterization of minimal linear languages.

Furthermore, by adding additional pairs to J_2 we can delimit the defined canonical language A in such a way that $\{f : f \in F(X^+)$ and for some g, $fg \in A$ and $g \in F(Y)F(X)\}$ is an arbitrary regular event (instead of simply the free monoid on X^+, as above), so that $L = \alpha(A \cap D_{XY})$ will be an arbitrary linear language. Thus we have an independent definition of the notion "linear language". (Notice that these further restrictions on J_2 affect only the permitted transitions in the matrices along the main diagonal of (60).

In much the same way, we can give a general definition of "metalinear language". Thus, for example, consider the particular metalinear language generated by the grammar with the equations

$$\begin{aligned} \xi_i &= \xi_{i1}\xi_2 \\ \xi_1 &= e + \Sigma\{a\xi_1 b : a,b \in V_T\} \\ \xi_2 &= e + \Sigma\{a\xi_2 b : a,b \in V_T\}. \end{aligned} \tag{61}$$

In this case, the matrix for the underlying standard regular event would be

(62)

	X_1^+	X_1^-	X_2^+	X_2^-
X_1^+	U	U	0	0
X_1^-	0	U	U	0
X_2^+	0	0	U	U
X_2^-	0	0	0	U

Any metalinear language, and only these, will be based on a standard event with a matrix of essentially this kind (with, perhaps, additional restrictions along the main diagonal).

Propositions 1 and 2 thus provide for the possibility of very natural definitions of the full class of *CF* languages, and various subfamilies of this class, independently of the approach taken in preceding sections.

6. UNDECIDABILITY

6.1. In Post [36] it is shown that the following problem, known as the *Correspondence problem*, is recursively unsolvable. Where $\Sigma = \{(f_1,g_1), \ldots, (f_n,g_n)\}$ is a sequence of pairs of strings, let us say that a sequence $I = (i_1, \ldots, i_m)$ of integers $(1 \leq i_j \leq n)$ *satisfies* Σ if

$$f_{i_1} \cdots f_{i_m} = g_{i_1} \cdots g_{i_m}. \tag{63}$$

The correspondence problem is the problem of determining whether, given Σ, there is an index sequence that satisfies Σ. Notice that either Σ is satisfied by no index sequence or else by infinitely many, since if $(i_1, \ldots, i_m)$ satisfies Σ, then so does $(i_1, \ldots, i_m, i_1, \ldots, i_m)$. Post showed that there is no algorithm for determining, for arbitrary Σ, whether there is no index sequence satisfying Σ, or whether there are infinitely many, these being the only alternatives.

We can reformulate the correspondence problem directly in terms of minimal linear grammars. Given $\Sigma = \{(f_1,g_1), \ldots, (f_n,g_n)\}$, form $G(\Sigma)$ with the single non-terminal S and the defining equation:

$$S = a + f_1 S g_1 + \ldots + f_n S g_n, \tag{64}$$

where a is a symbol not in any of the f_i's or g_i's. Clearly there is an index sequence satisfying Σ just in case $G(\Sigma)$ generates a string $faf^{\sim}$. Or, to put it differently, let L_m be the "mirror-image" language consisting of all strings $faf^{\sim}$, $f \in F(V_T)$, and let $L(G(\Sigma))$ be the language generated by G. Then either there is no index sequence satisfying Σ, in which case $L_m \cap L(G(\Sigma))$ is empty; or there are infinitely many index sequences satisfying Σ, in which case $L_m \cap L(G(\Sigma))$ is infinite. From the unsolvability of the correspondence problem and the fact that L_m is generated by a linear grammar with one non-terminal, we conclude directly that:

UNDECIDABILITY THEOREM 1.
There is no algorithm for determining, given two CF grammars G_1 and G_2 generating L_1 and L_2 respectively, whether $L_1 \cap L_2$ is empty or infinite. This is true even where G_1 and G_2 are minimal linear grammars and where G_1 is a fixed particular grammar of L_m.

The problems of emptiness or finiteness of intersection are easily seen to be solvable for one-sided linear grammars, but we see that for the simplest grammars in our framework that go beyond regular events in generative capacity, these problems are no longer solvable.

This observation is generalized in Bar-Hillel, Perles, Shamir [2], where many problems concerning *CF* grammars are shown to be recursively unsolvable. In brief, their method is as follows. Let us limit V_T to the set $\{a, 0, 1\}$. Where $\Sigma = \{(f_1,g_1), \ldots, (f_n,g_n)\}$ is a set of pairs of strings in the vocabulary $\{0, 1\}$ (i.e., $f_i,g_i \in F\{0, 1\}$), let $L(\Sigma)$ be the set of all strings

$$10^{i_k} \ldots 10^{i_1} a f_{i_1} \ldots f_{i_k} a \tilde{g}_{j_l} \ldots \tilde{g}_{j_1} a 0^{j_1} 1 \ldots 0^{j_l} 1, \tag{65}$$

where $1 \leq i_1, \ldots, i_k, j_1, \ldots, j_l \leq n$.

More perspicuously, let us use $\bar{\imath} = 01^i$ as a code for the number i. Then a string of $L(\Sigma)$ is formed by selecting index sequences $I = (i_1, \ldots, i_k)$ and $J = (j_1, \ldots, j_l)$ and forming

$$\overline{i_k} \ldots \overline{i_1} a f_{i_1} \ldots f_{i_k} a g_{j_l} \ldots g_{j_1} \overline{a j_1} \ldots \overline{j_l}. \tag{66}$$

$L(\Sigma)$ now plays the same role as the language generated by (64) in the foregoing proof of Undecidability Theorem 1. It is clearly a *CF* language (generated, in fact, by a meta-linear grammar which is an obvious modification of (64). But from *Theorem* 3, § 4.3, above, it follows directly that the complement $F(V_T) \setminus L(\Sigma)$ of $L(\Sigma)$ with respect to the vocabulary V_T is a *CF* language, and that we can construct its grammar given the grammar of $L(\Sigma)$. (Notice, in fact, that we could have used any code, in place of the particular choice $\bar{\imath} = 01^i$, for defining $L(\Sigma)$).

In place of the mirror-image language L_m used in the proof of *Undecidability Theorem 1*, let us consider the "double-mirror-image" language $L_{\hat{d}m}$ consisting of all strings

$$x_1 a x_2 a \tilde{x}_2 a \tilde{x}_1, \text{ where } x_1 \text{ and } x_2 \text{ are strings in } \{0, 1\}. \tag{67}$$

It is not hard to show that $L_{\hat{d}m}$ and its complement with respect to V_T are both *CF* languages.

Observe that

$$L(\Sigma) \cap L_{\hat{d}m} = \{\overline{i_k} \ldots \overline{i_1} a f_{i_1} \ldots f_{i_k} a g_{i_k} \ldots g_{i_1} a \overline{i_1} \ldots \overline{i_k}\} \tag{68}$$

where $(i_1, \ldots, i_k)$ satisfies Σ (that is, where $f_{i_1} \ldots f_{i_k} = g_{i_1} \ldots g_{i_k}$).

Observe also that an infinite set of strings of the form of (68) cannot constitute a *CF* language (nor, a fortiori, a regular event).

Suppose now that there is a positive solution to the correspondence

problem for Σ; that is, there is an index sequence satisfying Σ. Then, as we have observed, there are infinitely many such sequences. Consequently $L(\Sigma) \cap L_{\hat{d}m}$ is infinite. It is therefore neither a regular event nor a *CF* language.

Suppose, on the other hand, that there is no index sequence satisfying Σ. Then $L(\Sigma) \cap L_{\hat{d}m}$ is empty, and is therefore both a regular event and a context-free language. But $L(\Sigma)$ and $L_{\hat{d}m}$ are *CF* languages; and, with Σ fixed, we can construct their *CF* grammars $G(\Sigma)$ and $G_{\hat{d}m}$ (which are, in fact, meta-linear). Thus if there were an algorithm for determining whether the intersection of the languages generated by two *CF* grammars G_1 and G_2 is empty, finite, a regular event, or a *CF* language, this algorithm would also provide a solution to the general correspondence problem. We conclude, then:

UNDECIDABILITY THEOREM 2.
There is no algorithm for determining, given CF grammars G_1 and G_2, whether the intersection of the languages that they generate is empty, finite, a regular event, or a CF language — in particular, this remains true when both are meta-linear and G_2 is a fixed grammar of $L_{\hat{d}m}$.

Let $\overline{G}_{\hat{d}m}$ be the *CF* grammar that generates the complement $\overline{L_{\hat{d}m}}$ (all complements now are with respect to V_T) of $L_{\hat{d}m}$. And, given Σ, let $\overline{G}(\Sigma)$ be the *CF* grammar that generates the complement $\overline{L(\Sigma)}$ of $L(\Sigma)$, as guaranteed by *Theorem 3*, § 4.3. Consider now the grammar G generting the language $L(G) = \overline{L_{\hat{d}m}} \cup \overline{L(\Sigma)}$. Clearly G is *CF* and can be constructed from $G_{\hat{d}m}$ and $\overline{G}(\Sigma)$. But the complement $\overline{L(G)}$ of $L(G)$ is just the set $\overline{L_{\hat{d}m} \cup L(\Sigma)} = L_{\hat{d}m} \cap L(\Sigma)$, and we know by *Undecidability Theorem 2* that there is no algorithm for determining, given Σ, whether this set is empty, finite, a regular event, or a *CF* language. But given Σ, G is determined as a *CF* grammar. Therefore we have:

UNDECIDABILITY THEOREM 3.
There is no algorithm for determining, given the CF grammar G, whether the complement of the language generated by G is empty, finite, a regular event, or a CF language.

There is, in particular, no general procedure for determining whether the *CF* grammar G generates the universal language $F(V_T)$, or whether G generates a regular event (since the complement of a regular event is a

regular event). Consequently, there is no algorithm for determining, given *CF* languages L_1 and L_2, whether there is a transducer mapping L_1 onto L_2 since all and only regular languages can be obtained by transduction from the *CF* language $F(V_T)$ (Ginsburg and Rose, personal communication). There is, furthermore, no general method for determining whether two *CF* grammars are equivalent, i.e., generate the same language, since if there were such a method, it could be used to determine whether a *CF* grammar G is equivalent to the grammar G_U generating $F(V_T)$. It also follows immediately that there is no algorithm for determining, given *CF* grammars, whether the language generated by one includes the language generated by the other, since this would give a solution for the equivalence problem.

These results have been outlined for languages constructed from a three-element vocabulary V_T, but it is clear that by appropriate recoding, they still apply to languages in a vocabulary of two or more letters. This is worked out in detail in Bar-Hillel, Perles, Shamir [2].

6.2. We observed in § 4 that finite processes involving pairs of strings receive a natural formulation in terms of linear grammars. In particular, as we have just seen, the correspondence problem can be described directly as a problem concerning minimal linear grammars. The same is true of a second combinatorial problem, also due to Post, called the "Tag problem".

We can state a generalized form of the Tag problem in the following way. Let W be the set of strings (the free monoid) in some finite vocabulary, and let P be a finite subset of non-null strings of W meeting the condition that no string of W has more than one left factor in P. That is, there are no p_1, p_2, w_1, w_2, w_3 ($p_i \in P$, $w_i \in W$) such that $p_1 \neq p_2$ and $w_1 = p_1 w_2 = p_2 w_3$. Let V be the set of strings of W that have no left factor in P — that is, $v \in V$ if and only if there is no $p \in P$ such that $v = pw$, for $w \in W$. Clearly V is a recursive, in fact regular, set. Let α be a mapping of P into W (thus α defines a set of pairs of strings (p, w), where $w = \alpha p$, $p \in P$, $w \in W$). Define a mapping T on W, where

$$\begin{aligned} &Tf = f'\alpha p, \text{ if } f = pf' \\ &Tf = H, \text{ if } f \in V\ (H \notin W). \end{aligned} \tag{69}$$

Consider the problem:

(70) given a string f, is there an integer n such that $T^n f = H$?

Regarding T as defining the computation of a Turing machine, (70) is the *halting problem* for this Turing machine. It has been shown by Minsky [30 that (70) is a recursively unsolvable problem.

The Tag problem as formulated by Post is the special case of (70), above, where T meets the following additional conditions: P is the set of all strings of length k, for some fixed $k \geq 2$; αp depends only on the left-most symbol of p. Even with this restriction, the problem (70) is unsolvable, as Minsky has shown. This is a somewhat surprising result, because of the determinacy (*monogenicity*) of the generative procedure T.

As a step towards reformulating the generalized Tag problem in terms of minimal linear grammars, we observe that it can be stated in the following way. Given W, P, V, α, T, as above, the question (70) has a positive answer just in case

(71) there are strings $p_1, \ldots, p_n \in P$ and $v \in V$ such that:
$p_1 \ldots p_n v = f\alpha p_1 \ldots \alpha p_n$.

But we can now restate the generalized Tag problem as the following problem concerning linear grammars. Given W, P, V, α, T, let us define the grammar G generating $L(G)$ with the single equation

$$S = \Sigma_i v_i c + \Sigma_i (p_i S \widetilde{\alpha p_i}) \tag{72}$$

where $v_i \in V$, $p_i \in P$, and $c \notin W$ is the distinguished central marker. Let us define the language $M(f) = \{fgc\tilde{g} : g \in W\}$ (thus $M(f) = fL_m$, where L_m is the "mirror-image language" defined above). Then the answer to (71) (equivalently, (70)) is positive if and only if the intersection of $L(G)$ with $M(f)$ is non-empty. Thus we see that there is no algorithm for determining whether, for fixed f, the language $M(f)$ has a non-empty intersection with a language with a grammar meeting (72) (even for the special case in which P is the set of all strings of length k, for fixed $k \geq 2$, and αp depends only on the left-most letter of P).

Notice that *Undecidability Theorem 1*, above, also follows directly from unsolvability of the Tag problem. In fact, the Correspondence and Tag problems both concern the cardinality of the intersection of a minimal linear language L with the languages $M(f)$, where $f = e$ and L is arbitrary, for the case of the Correspondence problem, while f is arbitrary and L meets the condition (72), above, for the case of the Tag problem.

7. Ambiguity

7.1. We have defined the power series r to be *characteristic* just in case each coefficient $\langle r,f\rangle$ is either zero or one. We say that a *CF* grammar is *unambiguous* if the principal term of its solution is a characteristic power series. In this case, each sentence that it generates is provided with a single structural description, and "debracketization" introduces no ambiguities. Let us call a *CF* language *inherently ambiguous* if each of its *CF* grammars is ambiguous.

It is well-known that no regular event is inherently ambiguous — that is, each regular event is the support of a characteristic power series which is the principal term of the solution of a one-sided linear grammar [14], [37]. However, this remark does not carry over to the full class of *CF* grammars. It has been shown by Parikh [34] that there are *CF* languages that are inherently ambiguous.

An example of an inherently ambiguous language is the set

$$\{a^n b^m c^p : n = m \text{ or } m = p\}. \tag{73}$$

In this case, the strings of the form $a^n b^n c^n$ must have ambiguity at least two in any *CF* grammar generating (73) (and there is a *CF* grammar generating (73) in which they have ambiguity exactly two).

We do not have examples illustrating the extent of inherent ambiguity in *CF* languages, or special types of *CF* languages.

Notice that it is an immediate consequence of *Undecidability Theorem 1* of § 6 that there can be no algorithm for determining whether a *CF* grammar, or even a linear grammar, is ambiguous. Suppose in fact that, as above, $\Sigma = \{(f_1, g_1), \ldots, (f_n, g_n)\}$ is a sequence of pairs of strings. Select $n + 1$ new symbols $x_0, \ldots, x_n$ and construct the grammars G_f with the rules $S_f \to x_0$, $S_f \to x_i S_f \tilde{f}_i$ $(1 \leq i \leq n)$ and G_g with the rules $S_g \to x_0$, $S_g \to x_i S_g \tilde{g}_i$ $(1 \leq i \leq n)$. Clearly G_f and G_g are unambiguous, and the Correspondence problem for Σ has a positive solution if and only if there is a string generated by both G_f and G_g, that is, if and only if the grammar G_{fg} is ambiguous, where G_{fg} contains the rules of G_f, the rules of G_g, and the rules $S \to S_f$, $S \to S_g$, where S is the initial symbol of G_{fg}. Consequently, there can be no procedure for determining, for arbitrary Σ, whether the grammar G_{fg} associated with Σ in this way is unambiguous.

The grammar G_{fg} is linear with three non-terminals and a designated central marker, and we see that for this class of grammars the ambiguity

problem is unsolvable. Presumably, this remark can be generalized to grammars with two non-terminals. It is an interesting open question, however, whether the ambiguity problem remains unsolvable for minimal linear grammars.

Summarizing the matter of ambiguity, as it stands at present, we have the following results:

AMBIGUITY THEOREM 1. *There are inherently ambiguous CF languages.*

AMBIGUITY THEOREM 2.
There is no algorithm for determining whether a CF grammar (which may even be linear with a designated central marker) is ambiguous.

8. FINITE TRANSDUCTION

We want to describe a particularly simple family of transformations from language to language. The first and most essential one is a *homomorphism.*

Let L be any language on a terminal vocabulary Z and assume that for each $z \in Z$ we are given a language L_z on a second vocabulary X. We denote by θL the set of all strings (in X) which can be obtained by taking a word $g = z_{i_1} z_{i_2} \dots z_{i_m} \in L$, and replacing each z_{i_j} by an arbitrary word from $L_{z_{i_j}}$. The name "homomorphism" is self-explanatory. In fact, if we consider the rings $A(Z)$ and $A(X)$ of formal power series in the variables $z \in Z$ and $x \in X$, and if we denote by θ the homomorphism of $A(Z)$ into $A(X)$ that is induced by the mapping $\theta_z =$ the formal power series associated with L_z, then θL is the support of the image by θ of the formal power series associated with L.

An interpretation within our previous framework can be given if L and the L_z's are *CF* languages. In this case, suppose that L is produced by the *CF* grammar G (with non-terminal vocabulary Y) and that each L_z is produced by the *CF* grammar G_z (with the set of non-terminals Y_z and the initial letter $y_{z,0}$). We assume that the sets Y_z are disjoint and we consider a *CF* grammar $\bar{G}$ with non-terminals $Y \cup Z \cup \bigcup_{z \varepsilon Z} Y_z$ consisting of the rules of G and of the G_z's and the rules $z \to y_{z,0}$ $(z \in Z)$. (More simply we identify each z with $y_{z,0}$). It is clear that $\bar{G}$ produces exactly θL.

We now generalize this construction to the following type of context

dependency: Let R_i ($i \in I$) and $R_{i'}$ ($i' \in I'$) be two finite families of regular events such that every $g \in F(Z)$ belongs to one and only one member of each family. Suppose also that for each triple ($z \in Z$, $i \in I$, $i' \in I'$), we have a language $L_{z,i,i'}$ in the vocabulary X.

Then for any $y = z_{j_1} z_{j_2} \dots z_{j_k}$ we replace each z_{j_k} by an arbitrary string from the language $L(z_{j_k}, i, i')$ where i and i' are determined by the condition that the string $z_{j_1} z_{j_2} \dots z_{j_{k-1}}$ is in R_i and the string $z_{j_{k+1}} \dots z_{j_{k+1}}$ in $R_{i'}$. It is easily proven that without loss of generality it may be assumed that for any string g belonging to some set R_{i_1}, and for $z \in Z$, the set R_{i_2} which contains gz depends only upon the index i_1 and the letter z. In other words we may assume that we are given a set of states I, a transition mapping $I \times Z \to I$ and an initial state $i_0 \in I$ such that $z_{j_1} z_{j_2} \dots z_{j_{k-1}} \in R_i$ if and only if i is the state reached from i_0 after reading $z_{j_1} z_{j_2} \dots z_{j_{k-1}}$.

A similar construction applies to R_i, and for the sake of clarity we write the corresponding mapping as a *left* multiplication. Given the two mappings $I \times Z \to I$ and $Z \times I'$, we denote by σg, for each $g = z_{j_1} z_{j_2} \dots z_{j_m} \in F(X)$, the sequence of triples

$$(74) \qquad (i_1, z_{j_1}, i'_m)(i_2, z_{j_2}, i'_{m-1}) \dots (i_k, z_{j_k}, i'_{m-k+1}) \dots (i_m, z_{j_m}, i'_1)$$

where inductively

$$(75) \qquad i_2 = i_1 z_{j_1}, i_3 = i_2 z_{j_2}, \dots, i_m = i_{m-1} z_{j_m}, i_k = i_{k-1} z_{j_{k-1}} \text{ and}$$
$$i'_2 = z_{j_m} i'_1, i'_3 = z_{j_{m-1}} i'_2, \dots, i'_m = z_{j_2} i'_{m-1}.$$

With these notations the transformation we have been describing can be considered as consisting of two steps:

(76) (i) replacement of every $g \in L$ by the string $\sigma g = (i_1, z_{j_1}, i_m) \dots (i_m, z_{j_m}, i')$ in an alphabet U consisting of triples (i, z, i');

(ii) replacement in σg of every triple $(i_k, z_{j_k}, i'_{m-k+1})$ by an arbitrary string from the language $L(z_{j_k}, i_k, i'_{m-k+1})$.

Since step 2 is only a homomorphism, it is sufficient to discuss step 1. For this let U denote the set of all triples (i, z, i') and consider the language L' obtained from L by adding to its grammar all the rules $z_j \to (i, z_j, i')$ with $i \in I$, $i' \in I'$ arbitrary).

Clearly a string of L' belongs to the set $\{\sigma g : g \in L\}$ if and only if it satisfies the condition (75) above, or, in other words, if it belongs to the

regular event $\bar{R}$ determined by the condition (75) on the set $F(U)$ of all strings in the alphabet U.

Hence step 1 consists only of a homomorphism from L in to the set of all strings on U (which gives L') followed by the intersection of L' with a regular event.

Let us now give a final interpretation of what we have done: For each $z \notin Z$, let μz denote a matrix whose rows and columns are indexed by pairs $(i \notin I, i' \notin I')$ and whose entries are as follows

$$\mu z_{(i,\, i')(i''\, i''')} = \text{the triple } (i, z, i''') \text{ if } i'' = iz \text{ and } i' = zi''' \\ = 0 \text{ otherwise.} \tag{77}$$

Then if we compute
$\mu z_{j_1} \mu z_{j_2} \dots \mu z_{j_m} = \mu g$, it is easily verified that the entry $(i, i'_m)(i_m, i'_1)$ of μg is precisely σg. From this it follows easily that $\{\sigma g : g \notin L\} = L' \cap \bar{R}$ is also a context-free language. Indeed, μ is a homomorphism — we replace every non-terminal y by a matrix μy whose entries are new non-terminals and we verify that μ commutes with the substitutions used for defining the language as the solution of a system of equations. On the other hand identifying the entries one by one in the image μ of our equations gives a new set of equations of the usual type that exactly defines $L' \cap \bar{R}$ [46]. More simply still we can define μ' as above except that for each non-zero entry we take the formal power series associated with $L(z_j, i, i')$, instead of the triple (i_1, z_j, i'). Then the two steps of the construction are telescoped in a single one and the power series associated with the language (on X) obtained by our transformation is simply an entry of

$$\Sigma\{\mu' g : g \in L\}. \tag{78}$$

This is the basis for the proof of Theorem 2, § 4, above.

9. Connections with the Theory of Automata

We have so far been studying generative processes, the languages and systems of structural descriptions that they define, and finitary mappings on these languages from a completely abstract point of view. To relate these remarks to the theory of automata, it is convenient to introduce a temporal asymmetry into consideration.

An automaton M can be regarded as a device consisting of a set of states Σ (the *memory* of M) that accepts (equivalently, produces) a

sequence of symbols from a vocabulary (alphabet) V in accordance with fixed, finitely statable instructions (which can be given by associating with each $v \in V$ a mapping φ_v of Σ into itself (or into the set of subsets of Σ, in the case of a "non-deterministic" automaton). If we designate an initial state and a set of final states, we define a language $M(L)$ consisting of the strings that can be accepted by M as it proceeds in accordance with its instructions from the initial state to a final state, proceeding from $S \in \Sigma$ to $S' \in \Sigma$ on accepting v just in case $\varphi_v(S) = S'$ (or $S' \in \varphi_v(S)$, in the non-deterministic case). The size of memory of M, or its rate of growth in the course of computation, provides a certain index of the richness of the language $L(M)$ in terms of which we can compare various families of languages of the kinds we have considered.

Given a set of strings L, let us write $f \sim f'$ just in case for all g, $fg \in L$ if and only if $f'g \in L$. Clearly $\sim$ is an equivalence. Furthermore, it is clear that we can take the equivalence classes defined by $\sim$ as states of an automaton $M(L)$ that accepts L, since all of the information about f relevant to the further computation of $M(L)$, once it has read f, is given by the equivalence class to which f belongs. Notice that L is the union of certain of these equivalence classes, and that $f \sim f'$ implies that $fg \sim f'g$, for all g.

Secondly, given L let us write $f \equiv f'$ if and only if for all g, $gf \sim gf'$. Clearly $f \equiv f'$ if and only if for all g,g', gfg' if and only if $gf'g'$. Thus $\equiv$ is symmetrical, and it is easy to show that $f_1 \equiv f_2$ and $f_3 \equiv f_4$. Thus $\equiv$ is a congruence relation, and the $\equiv$-classes in the set $F(V)$ can be multiplied together giving a quotient monoid of $F(V)$. This quotient monoid $F'(V) = \varphi F(V)$ is such that $L = \varphi^{-1}\varphi L$ — and is canonically associated with L_0 [41].

This observation relates the present theory to the theory of monoids. The interest of this is that in certain cases, the $\sim$-classes (and the quotient monoid) have a simple interpretation that can be translated into the language of automata, and, conversely, that certain algebraic notions (in particular, that of *extension*), receive a simple interpretation.

Returning now to the problem of characterizing families of languages in terms of automata, it is well-known that the sub-family $\mathrm{Sup}(\mathscr{L}_0{}^+)$ of *CF* languages is uniquely characterized by the fact that for each language $L \in \mathrm{Sup}(\mathscr{L}_0{}^+)$, there is an automaton $M(L)$ with bounded memory that accepts L.

Consider now the family $\mathrm{Sup}(\mathscr{L}_0)$, that is, the set of supports of power series that are the solutions to systems of "one-sided linear" equations

with positive or negative integral coefficients. As we have observed, $L \in \mathrm{Sup}(\mathscr{L}_0)$ if and only if $L = \mathrm{Sup}(r_1 - r_2)$, where $r_1, r_2 \notin (L_0^+)$. It can now be shown that the following statements are equivalent:

(79) (i) $L \in \mathrm{Sup}(\mathscr{L}_0^+)$;
(ii) there is a one-one correspondence between the $\sim$-classes for L and a finite dimensional space of integral vectors $v(f)$ such that for each $x \in V$, $v(fx) = v(f)\mu x$, where μx is a matrix;
(iii) $F/\equiv$ is isomorphic to a monoid of finite dimensional integral matrices (i.e., the matrices μ of ii);
(iv) L is accepted by an automaton $M(L)$ with a finite dimensional space of vectors with integral coordinates as memory and transitions as above in ii.

(Schützenberger, [44] — let the class $\mathscr{A}$ of automata be those defined by (79iv)).

Consider now the following two restrictions on the class $\mathscr{A}$ of automata.

(80) (i) there is an N such that, for all $f \in F(V)$, $||v(f)|| < N$;
(ii) for all $f, f', f'' \in F(V)$ and $\varepsilon > 0$,
$$\lim_{n \to \infty} e^{-\varepsilon n} ||v(f'f^n f'')|| = 0$$

where $||v||$ is the length of the vector v, in the usual sense.

Clearly (80i) implies (80ii). Furthermore, it is clear that L is a regular event (that is, $L \in \mathrm{Sup}(\mathscr{L}_0^+)$) just in case (80i) is met by an automaton of class $\mathscr{A}$ that accepts L. An automaton of class $\mathscr{A}$ that meets condition (80ii) is called a *finite counting automaton* in Schützenberger [47], where such devices are studied. It can be proved that in a loose way (80ii) means that the amount of information (in bits) stored in the memory does not grow faster than a linear function of the logarithm of the length of one input word.

It is interesting to observe that (just as in the case of the full class of *CF* grammars), there is no algorithm for determining, given $M \in \mathscr{A}$, whether there is an f not accepted by M [28]. Furthermore, the same problem for finite counting automata is easily shown to be unsolvable, if Hilbert's tenth problem (the problem of the existence of an integral solution for an arbitrary diophantine equation) is unsolvable [47].

Consider now an automaton M with a structure of the following kind: the states of M (the $\sim$-classes in the input language of M) are identified with strings in a certain new ("internal") alphabet, and for

each $v \in V$, the "computing instruction" φ_v mapping [$\sim$-class of f] $\to$ [$\sim$-class of fv] consists of addition or deletion of letters at the right-hand end of the internal string associated with [$\sim$-class of f]. Such an automaton we can call (in accordance with usual terminology) a *pushdown storage (PDS) automaton.* PDS automata constitute a restricted subclass of the class of linear bounded automata studied by Myhill [319], Ritchie [39].

Where M is a PDS automaton, the language L that it accepts is a *CF* language, and each *CF* language can be obtained by a homomorphism from a language accepted by a PDS automaton [48], [49]. In particular, where D is a Dyck language and A a standard regular event (cf. § 5), $D \cap A$ is accepted by a PDS automaton.

A *non-deterministic PDS automaton* is an automaton of the type described above, except for the fact that φ_v maps a state into a set of states. We can now prove directly that *CF* languages (languages of the class $\mathrm{Sup}(\mathscr{S}^+)$) are exactly those that are accepted by non-deterministic PDS automata [11], [12].

REFERENCES

[1] AJDUKIEWICZ, K., Die Syntaktische Konnexität. *Studia Philosophica*, (1935), 1, 1–27.

[2] BAR-HILLEL, Y., PERLES, M. and SHAMIR, E., On formal properties of simple phrase structure grammars. *Tech. Report* 4, July 1960.

[3] —, Applied Logic Branch. The Hebrew University of Jerusalem. Now published in *Zeit. für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, Band 14, Heft 2, (1961), 143–172.

[4] — and SHAMIR, E., Finite state languages, *Bull. Research Council Israel*, 8F (1960), 155–166.

[5] BIRKELAND, R., Sur la convergence de développement, qui expriment le racins de l'équation algebrique générale. *C. R. Acad. Sciences* 171 (1920) 1370–1372; 172, (1921) 309–311.

[6] CULIK, K., Some notes on finite state languages. *Časopis pro pěstovani Mat.*, (1961), 86, 43–55.

[7] CHOMSKY, N., Three models for the description of language. *I.R.E. Trans.* PGIT 2, (1956), 113–124.

[8] —, On certain formal properties of grammars. *Information and Control*, (1959), 2, 137–167.

[9] —, A note on phrase structure grammars. *Information and Control*, (1959), 2, 393–395.

[10] —, On the notion "Rule of Grammar". *Proc. Symp. Applied Math. 12, Am. Math. Soc.*, (1961).

[11] —, Context-free grammars and pushdown storage. Quarterly Progress Reports no. 65, Research Laboratory of Electronics, M.I.T., (1962).

[12] —, Formal properties of grammars 1962. To appear in Bush, Galanter, Luce (eds.), *Handbook of Mathematical Psychology*, vol. 2. Wiley.

[13] —, The logical basis for linguistic theory. *Proc IXth Int. Cong. Linguists*, Cambridge, Mass. (1962).

[14] — and MILLER, G. A., Finite state languages. *Information and Control*, 1 (1958), 91–112.

[15] — and —, Introduction to the formal analysis of natural languages 1962. To appear in Bush, Galanter, Luce (eds.), *Handbook of mathematical psychology*, vol. 2, Wiley.

[16] DAVIS, M., *Computability and Unsolvability*. New York, McGraw-Hill, (1958).

[17] ELGOT, C. C. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98 (1961), 21–51.

[18] GINSBURG, S. and RICE, H. G., Two families of languages related to ALGOL. *Technical Memorandum.* Systems Development Corporation; Santa Monica, California, (1961).

[19] — and ROSE, G. F., Operations which preserve definability in languages. *Technical Memorandum.* Systems Development Corporation. Santa Monica, California, (1961).

[20] JUNGEN, R., Sur les séries de Taylor n'ayant que des singularités algebrico-logarithmiques sur leur cercle de convergence. *Comm. Math. Helvetici*, 3 (1931), 286–306.

[21] KLEENE, S. C., Representation of events in nerve nets and finite automata. *Automata Studies*, Princeton University Press, (1956), 3–41.

[22] KULAGINA, O., Ob odnom sposobe opredelenija grammatičeskix ponjatij. *Problemy Kibernetiki*, 1, Moscow, (1958).

[23] LAMBEK, J., The mathematics of sentence structure. *Am. J. Math.*, 65 (1958), 153–170.

—, On the calculus of syntactic types. *Proc. Symposium Applied Math.* 12, Am. Math. Soc., (1961).

[25] LYNDON, R. C., Equations in free groups, *Trans. Am. Math. Soc.* 96 (1960), 445–457.

[26] MCNAUGHTON, R., The theory of automata. To appear in *Advances in Computers*, vol II (Academic Press).

[27] MAHLER, K., On a theorem of Liouville in fields of positive characteristic. *Canadian J. of Math.* 1, (1949), 397–400.

[28] MARKOV, A. A., Ob odnoi nevazrešimoi probleme, *Doklady Akad. Nauk:* n.s. 78, (1951), 1089–1092.

[29] MILLER, G. A. and CHOMSKY, N., Finitary models of language users. 1962. To appear in Bush, Galanter, Luce (eds.), *Handbook of Mathematical Psychology*, vol. 2, Wiley.

[30] MINSKY, M. L., Recursive unsolvability of Post's problem of Tag. *Ann. of Math.*, 74, (1961), 437–455.

[31] MYHILL, J., Linear bounded automata. WADD *Tech. Note* 60–165. Wright Air Dvpt. Division. Wright Patterson Air Force Base Ohio, (1960).

[32] NEWELL, A. and SHAW, J. C., Programming the logic theory machine. *Proc. Western Joint Computer Conference*, (1957), 230.

[33] OETTINGER, A. G., Automatic syntactic analysis and the pushdown store. *Proc. of Symposia in Applied Math.*, 12, Am. Math. Soc., (1961).

[34] PARIKH, R. J., Language generating devices. *Quarterly Progress Report* no. 60. Research Laboratory of Electronics, M.I.T. January (1961), pp. 199–212.

[35] PERLES, M., RABIN, M. O. and SHAMIR, E., The theory of definite automata. *Tech. Report* no. 6, O.N.R., (1961).

[36] POST, E., A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, (1946), 52, 264–268.

[37] RABIN, M. O. and SCOTT, D. Finite automata and their decision problems. *I.B.M. Journal of Research*, 3, (1959), 115–125.

[38] RANEY, G. N., Functional composition patterns and power-series reversion, *Trans. Am. Math. Soc.*, 94 (1960), 441–451.

[39] RITCHIE R. W., *Classes of recursive functions of predictable complexity*. Doctoral Diss, Dept. of Math, Princeton U, (1960).

[40] SCHEINBERG, S., Note on the Boolean properties of context-free languages. *Information and Control*, 3 (1960), 372–375.

[41] SCHÜTZENBERGER, M. P., On an application of semi-group methods. *I.R.E. Trans.*, IT2, (1956), 47–60.

[42] —, Un problème de la théorie de automates. *Séminaire Dubreil-Pisot* (Paris) Dec., (1959).

[43] —, A remark on finite transducers. *Information and Control*, 4 (1961), 185–196.

[44] —, On the definition of a family of automata. *Information and Control*, 4 (1961), 245–270.

[45] —, Some remarks on Chomsky's context-free languages. *Quarterly Progress Report* no. 68, Research Laboratory of Electronics, M.I.T., Oct. (1961).

[46] —, On a theorem of R. Jungen, 1962. To appear in *Proc. Am. Math. Soc.*

[47] —, Finite counting automata, 1962. To appear in *Information and Control.*

[48] —, On Context-free languages and pushdown storage. To appear in I.B.M. *Journal of Research.*

[49] —, Certain elementary families of automata. *Symp. on mathematical theory of automata*, Polytechnic Institute of Brooklyn, 1962.

[50] SHEPHERDSON, J. C., The reduction of two-way automata to one-way automata. I.B.M. *Journal of Research*, (1959), 198–200.

[51] SHAMIR, E., On sequential languages. *Tech. Report* no. 7, O.N.R., (1961).

[52] YAMADA, A., Counting by a class of growing automata. *Doctoral Diss.*, Univ. of Penna., Philadelphia, (1960).