JOURNAL OF COMPUTER AND SYSTEM SCIENCES 5, 397–429 (1971)

Time-Bounded Grammars and Their Languages*

RONALD V. BOOK

Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138

Received November 20, 1970

Formal grammars and formal languages are studied from the viewpoint of time-bounded grammars. A time-bound on a grammar is a measure of the "derivational complexity" of the language generated. Based on results of Gladkii [7] on connectivity in grammars, it is shown that a "linear speedup" can be obtained and that one can construct Turing acceptors to simulate grammars without loss of time. Positive containment and closure properties are also studied.

INTRODUCTION

The subject of this paper is the study of formal grammars and the study of formal languages from the viewpoint of time-bounded grammars. Much recent activity in automata theory is concerned with the computational complexity of formal languages as measured by the recognition capacity of time-bounded or space-bounded Turing acceptors. Here families of languages are defined by placing bounds on the time function of grammars—this represents a measure of the "derivational complexity" of the language generated by the grammar. The various properties of such grammars and their languages are then studied.

The "time function" T_G of a grammar $G = (V, \Sigma, R, X)$ was first defined by Gladkii [7]. For any positive integer n such that G generates a string of length n, $T_G(n)$ is the maximum value of the set of numbers $\{t_G(w) \mid X \overset{*}{\Rightarrow} w \text{ in } G \text{ and } w \text{ has length } n\}$, where for any string $w \in V^*$ such that $X \overset{*}{\Rightarrow} w$, $t_G(w)$ is the length of the minimum length derivation of w. If G generates no such string, $T_G(n)$ is undefined. A nondecreasing total recursive function f "bounds" the grammar G, if there exists

* This paper represents a portion of the author's Ph.D. thesis at Harvard University [1]. Some of these results were announced at the IEEE Ninth Annual Symposium on Switching and Automata Theory, Schenectady, New York, October, 1968. This research was supported in part by Air Force Cambridge Research Laboratories, Air Force Systems Command, USAF, under Contract F19628-68-C-0029, and by the National Aeronautics and Space Administration under Grant No. NGR 22-007-176.

$k > 0$ such that for all $n \geqslant k$, if $T_G(n)$ is defined, then $T_G(n) \leqslant f(n)$. For certain "bounding functions" f, we define the families $\mathscr{L}_{\text{CS}}(f) = \{L(G) \mid G$ is a CS grammar such that f bounds $G\}$ and $\mathscr{L}(f) = \{L(G) \mid G$ is an arbitrary grammar (possibly with erasing rules) such that f bounds $G\}$. These are the families of languages studied in this paper.

Certain questions arise immediately. (1) If f is a bounding function and $k > 0$ a constant, how do $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}_{\text{CS}}(kf)$ compare? Similarly, how do $\mathscr{L}(f)$ and $\mathscr{L}(kf)$ compare? In light of the result of Hartmanis and Stearns [13] for time-bounded multitape Turing machines, one suspects the families are equal. (2) If f is a bounding function and $L \in \mathscr{L}_{\text{CS}}(f)$ or $L \in \mathscr{L}(f)$, does there exist some type of Turing acceptor which accepts L and which accepts within time bound f? (3) What closure properties do the families possess?

The first two questions are answered in the affirmative by applying a theorem on "connectivity" in grammars. Informally, a derivation is connected if at each step the part of the string to which the rewriting rule is applied contains part of the result of the application of the rule applied at the previous step. It is shown that for every grammar G, one can effectively construct a grammar G' such that $L(G') = L(G)$, such that every proper derivation of G' is connected, and such that $T_{G'}$ is bounded by a constant multiple of T_G. This result (Theorem 3.4), a modification of a result of Gladkii [7], yields a "linear speedup" just as for Turing acceptors (Theorem 3.5). It is applied again to show that a nondeterministic multitape Turing acceptor can imitate the derivations in a grammar without loss of time (Theorem 4.2).

Corollaries to the result on connectivity become useful in studying the positive closure properties of the families $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$. It is shown that if mild restrictions are placed on the bounding functions, then these families are AFLs [6] closed under reversal and e-free substitution. If a bounding function is of the order of x^2 or greater, then the families are closed under intersection, but this is not the case if x^2 majorizes the bounding function.

Particular attention is paid to the family $\text{LINEAR}_{\text{CS}}$ of languages generated by context-sensitive grammars bounded by linear functions. It is shown that $\text{LINEAR}_{\text{CS}}$ is a subfamily of every $\mathscr{L}_{\text{CS}}(f)$, that $\text{LINEAR}_{\text{CS}}$ contains the e-free context-free languages but is a proper subfamily of the context-sensitive languages, and that the closure of $\text{LINEAR}_{\text{CS}}$ under arbitrary homomorphic mappings is the family of all recursively enumerable sets. From these results it is shown that most of the questions known to be undecidable for context-sensitive languages are also undecidable for the languages in $\text{LINEAR}_{\text{CS}}$ and thus in every $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$.

The paper itself is divided into seven sections. Section 1 contains definitions and notation from automata and formal language theory which are necessary for this paper. In Section 2, time functions, bounding functions, and the families $\mathscr{L}_{\text{CS}}(f)$, etc., are formally defined and their basic properties are investigated. Section 3 discusses connectivity and Section 4 contains results on the simulation of grammars by machines.

In Section 5, positive closure and containment results are studied, while negative closure and undecidability results are given in Section 6. Section 7 contains results relating $\text{LINEAR}_{\text{CS}}$ to other families of languages and other hierarchy results.

1. Basic Definitions and Notation

In this section we state some basic definitions of automata theory and establish notation. For background see [4] and [15].

A grammar is a quadruple $G = (V, \Sigma, R, X)$, where the *vocabulary* or *alphabet* V is a finite set of distinct symbols, $\Sigma \subset V$ is the *terminal* alphabet or vocabulary, $X \in V - \Sigma$ is the *initial* or *starting* symbol, and R is a finite set of *rewriting rules* or *productions* of the form $\alpha_1 y_1 \cdots \alpha_n y_n \alpha_{n+1} \rightarrow \alpha_1 w_1 \cdots \alpha_n w_n \alpha_{n+1}$ with each $\alpha_i \in \Sigma^*$, $y_i \in (V - \Sigma)^+$, $w_i \in V^*$, and for some i, $w_i \neq y_i$.[1] If $\rho \rightarrow \theta \in R$, then for any $\alpha, \beta \in V^*$, we write $\alpha\rho\beta \Rightarrow \alpha\theta\beta$ and say that the rule $\rho \rightarrow \theta$ is *applicable* to the string $\alpha\rho\beta$ and that $\rho \rightarrow \theta$ *transforms* $\alpha\rho\beta$ to $\alpha\theta\beta$. If $\theta_0, \theta_1, ..., \theta_n \in V^*$ and for each $i > 0$, $\theta_{i-1} \Rightarrow \theta_i$, then $D : \theta_0 \Rightarrow \theta_1 \Rightarrow \cdots \Rightarrow \theta_n$ is a *derivation* of *length n*. If $\theta, \theta' \in V^*$, we write $\theta \overset{+}{\Rightarrow} \theta'$ if for some $n > 0$, there is a derivation $D : \theta_0 \Rightarrow \cdots \Rightarrow \theta_n$ of length n such that $\theta = \theta_0$ and $\theta' = \theta_n$, and write $\theta \overset{*}{\Rightarrow} \theta'$ if $\theta' = \theta$ or $\theta \overset{+}{\Rightarrow} \theta'$. A derivation $D : \theta_0 \Rightarrow \cdots \Rightarrow \theta_n$ is *proper* if $\theta_0 = X$ and if $i \neq j$ implies $\theta_i \neq \theta_j$. For any set $U \subseteq V^*$, $L(G, U) = \{w \in U \mid \text{there is a proper derivation } X \Rightarrow \cdots \Rightarrow w \text{ in } G\}$. *The language generated by* G is $L(G) = L(G, \Sigma^*)$, so $L(G) = \{w \in \Sigma^* \mid \text{there is a proper derivation } X \Rightarrow \cdots \Rightarrow w \text{ in } G\}$.

A language $L \subseteq \Sigma^*$ is *e*-free if $e \notin L$. In this paper we shall deal exclusively with *e*-free languages, so that for any grammar G considered we have $L(G) = L(G, \Sigma^+)$.

A rule $\rho \rightarrow \theta$ is *length-preserving if* $|\rho| = |\theta|$, *decreasing* or *erasing* if $|\rho| > |\theta|$, and *increasing* if $|\rho| < |\theta|$.[2]

A grammar $G = (V, \Sigma, R, X)$ is *Type* 0 if each rule in R is of the form $\alpha Z \beta \rightarrow \alpha\gamma\beta$ where $\alpha, \beta, \gamma \in V^*$ and $Z \in V - \Sigma$.[3] It is well known that for any grammar G there is a Type 0 grammar G' such that $L(G') = L(G)$, and that the family of languages generated by all Type 0 grammars is the family of recursively enumerable sets.

A grammar $G = (V, \Sigma, R, X)$ is *monotonic* if $\rho \rightarrow \theta \in R$ implies $|\rho| \leqslant |\theta|$. A grammar is *context-sensitive* (CS) if it is both Type 0 and monotonic. A language L is a CS *language* if there is a CS grammar G such that $L(G) = L$. It is well-known that if G is a monotonic grammar, then $L(G)$ is a CS language [5, 16]. Denote the family of CS languages by $\underline{\text{CS}}$.

A grammar $G = (V, \Sigma, R, X)$ is *context-free* (CF) if $\rho \rightarrow \theta \in R$ implies $\rho \in V - \Sigma$,

[1] For a set A, A^* is the monoid with identity e freely generated by A. $A^+ = A^*A$.

[2] For a string w, $|w|$ is the length of w.

[3] A Type 0 grammar is sometimes called *context-sensitive with erasing.*

i.e., $| \rho | = 1$. A language L is a CF *language* if there is a CF grammar G such that $L(G) = L$. Denote the family of CF languages by $\underline{\text{CF}}$.

A grammar $G = (V, \Sigma, R, X)$ is *finite-state* if each rule in R is of the form $Z \rightarrow aY$ or $Z \rightarrow a$ or $Z \rightarrow e$, where $Z, Y \in V - \Sigma$, $a \in \Sigma$. A language L is a *finite-state language* if there is a finite-state grammar G such that $L(G) = L$. The family of finite-state languages is identical to the family of regular sets, those sets accepted by finite-state automata.

It is assumed that the reader is familiar with the basic concepts of automata theory. In particular, a basic knowledge of finite-state automata, pushdown store automata, linear bounded automata, and Turing machines is assumed. It is well known that a language is finite state if and only if it is accepted by a finite-state automaton (deterministic or nondeterministic), that a language is CF if and only if it is accepted by a nondeterministic pushdown store automaton, that a language is CS if and only if it is accepted by a linear bounded automaton, and that a language is generated by a Type 0 grammar if and only if it is accepted by a Turing machine. In this paper we consider acceptance by several types of Turing machines and we briefly review their definitions.

A *one-tape off-line* Turing machine has exactly one tape, (see [14]). Initially, the input is assumed to be written on the tape and a computation begins with the single read-write head scanning the leftmost input symbol. There is just one read-write head on this tape and this head can both read and write in any tape square. The transitions of this machine depend only on the current internal state of the machine and the contents of the tape square being scanned. In any one step of a computation of such a machine, any combination of the following operations may be performed: change the internal state; write in the tape square currently scanned (i.e., change the scanned symbol); move the read-write head either one square to the left or one square to the right or do not move the head at all; the machine halts.

An *on-line, multitape* Turing machine has one tape upon which the input is originally written. This tape has only one head and this is a read-only head. The input is scanned from left to right during a computation and the read-only head never moves left. This machine may also have some finite number of storage tapes, each storage tape having exactly one read-write head (see [13]). In any one step of a computation of such a machine, any combination of the following operations may be performed: change the internal state; move the read-only head of the input tape one square to the right; on each of the storage tapes, write in the currently scanned tape square (i.e., change the scanned symbol); on each of the storage tapes, move the read-write head one square to the left or one square to the right or do not more the read-write head; the machine halts.

If M is a Turing machine and f is a function, then *M accepts within time bound f* if for each input w accepted by M, there is a computation of M upon input w which accepts w and which has no more than $\max(|f(w)|, | w |)$ steps.

In most cases the Turing machines discussed are nondeterministic. Thus, at any step in a computation there is a finite set of possible transitions.

The set of positive integers is denoted by $\mathbb{N}$.

2. Time Functions and Bounding Functions

In this section we define the time function of a grammar. We also define bounding functions and the families of languages determined by bounding functions. Some basic properties of grammars, their time functions, and bounding functions are established.

We begin by developing the notion of the time function T_G of the grammar G. T_G is a (partial) function such that for any $n > 0$ for which $T_G(n)$ is defined, $T_G(n)$ bounds the length of the shortest derivations of all strings of length equal to n which are generated by G. This definition is due to Gladkii [7]. We first define the (partial) function t_G which assigns to a string w the length of the shortest derivation of w in G if such a derivation exists.

Definition 2.1. For any grammar $G = (V, \Sigma, R, X)$ and every $w \in L(G, V^+)$, let $t_G(w)$ be the least integer m such that there is a proper derivation of w in G of length m.

For any grammar $G = (V, \Sigma, R, X)$, $t_G : V^* \to \mathbb{N}$ is a partial recursive function. To see this, note that $L(G, V^+)$ is a recursively enumerable set since $L(G, V^+)$ can be enumerated by considering for each $n > 0$, the strings generated by all proper derivations of length no greater than n. If $X \Rightarrow \cdots \Rightarrow w$ is a proper derivation in G of length m, then $t_G(w) \leqslant m$. To compute $t_G(w)$ it is sufficient to consider all proper derivations of G of length no greater than m and find the shortest derivation of w.

Definition 2.2. For any grammar $G = (V, \Sigma, R, X)$, define $T_G : \mathbb{N} \to \mathbb{N}$ the (partial) *time function* of G by

$$T_G(n) = \begin{cases} \max\{t_G(w) \mid w \in V^n, X \overset{+}{\Rightarrow} w\}, & \text{if } L(G, V^n) \neq \phi \\ \text{undefined otherwise.} \end{cases}$$

The time function T_G of a grammar G may be viewed as a measure of "derivational complexity" [7], and one may measure the derivational complexity of a language L by assigning to L the "smallest" time function of all grammars which generate L. We now proceed to investigate properties of the functions T_G and to define families of languages based on "bounds" on derivational complexity.

In this paper we deal only with e-free languages. This is a technical convenience which simplifies certain proofs and the principal results will still hold if this restriction is relaxed (but certain arguments would need to be revised). Thus neither $t_G(e)$ nor $T_G(0)$ are defined.

If for some $G = (V, \Sigma, R, X)$, T_G is total recursive, then both $L(G, V^+)$ and $L(G)$ are recursive sets. In fact, if T_G is bounded above by a nondecreasing partial recursive function, then both $L(G, V^+)$ and $L(G)$ are recursive sets. However in general T_G need not be partial recursive. We see this in Proposition 2.4 below.

Notation. For any partial function $g:A \to B$, we write $g(a)\downarrow$ if $g(a)$ is defined and $g(a)\uparrow$ otherwise.

LEMMA 2.3. *Let $G = (V, \Sigma, R, X)$ be a grammar and let f be a nondecreasing partial recursive function. If for all n such that $T_G(n)\downarrow$, both $f(n)\downarrow$ and $T_G(n) \leqslant f(n)$, then $L(G, V^+)$ and $L(G)$ are recursive sets.*

Proof. Let k be the number of elements in V. Let $w \in V^+$ and let $m = |w|$. Consider the set S_m of all proper derivations in G of length no greater than $c(m) = \sum_{1 \leqslant i \leqslant m} k^i$. Since there are exactly $c(m)$ words in V^+ of length no greater than m, either every word in $L(G, V^+)$ has length strictly less then m or there is a derivation $X \Rightarrow \cdots \Rightarrow w'$ in S_m such that $|w'| \geqslant m$. Since S_m is a finite set, one can search S_m and find such a derivation and string w' or find that there are none. In the latter case, $w \notin L(G, V^+)$. In the former case, we have $w' \in L(G, V^+)$ and $T_G(|w'|)\downarrow$, and thus $f(|w'|)\downarrow$. Since f is nondecreasing and $m \leqslant |w'|$, then $T_G(m) \leqslant f(m) \leqslant f(|w'|)$. Thus $w \in L(G, V^+)$ if and only if there is a proper derivation of w in G of length no greater than $f(|w'|)$. Thus $L(G, V^+)$ is recursive, and since Σ^+ is recursive and $L(G) = L(G, V^+) \cap \Sigma^+$, $L(G)$ is recursive.

PROPOSITION 2.4. *There exists a grammar G such that T_G is not a partial recursive function.*

Proof. Let $G_0 = (V, \Sigma, R, X)$ be any grammar such that $L(G_0)$ is not recursive. Let A and Y be new symbols not in V, and let $V_1 = V \cup \{Y, A\}$. Let

$$R_1 = R \cup \{Y \to \theta \mid X \to \theta \in R\} \cup \{Y \to AY,\ Y \to A\}.$$

Let $G = (V_1, \Sigma, R_1, Y)$. Clearly, $L(G) = L(G_0)$ and T_G is a total function (since for any n, $Y \overset{+}{\Rightarrow} A^n$). Thus if T_G is partial recursive, then it must be total recursive and so $L(G)$ is recursive. But G_0 was chosen so that $L(G_0)$ is not recursive. Hence, T_G is not partial recursive.

We now consider a property of time functions which is useful in forming our definition of bounding functions. For $G = (V, \Sigma, R, X)$, let

$$k = \max\{|\theta| - |\rho| \mid \rho \to \theta \in R\}.$$

For any $w \in L(G, V^+)$, every derivation of w from X in G has length at least $|w|/k$ since one step in a derivation can increase the length of a string by at most k. Thus T_G is "at least linear", i.e., for any n such that $T_G(n)\downarrow$, $n/k \leqslant T_G(n)$. Thus we have the following proposition:

PROPOSITION 2.5. *If $G = (V, \Sigma, R, X)$ is a grammar such that $L(G, V^+)$ is infinite, one can effectively find a positive integer k such that for any n, if $T_G(n)\downarrow$, then $n/k \leqslant T_G(n)$.*

DEFINITION 2.6. A function f is a *bounding function* if it is a nondecreasing total recursive function with the property that there is a positive integer k such that for all x, $f(x) \geqslant x/k$, and such that for all $x \geqslant 0, f(x) \geqslant 0$.

DEFINITION 2.7. A bounding function f *bounds* grammar $G = (V, \Sigma, R, X)$ if for any integer, n, if $T_G(n)\downarrow$, then $T_G(n) \leqslant f(n)$.

DEFINITION 2.8. If f is a bounding function, define $\mathscr{L}_{\mathrm{CS}}(f) = \{L(G) \mid G \text{ is a CS grammar and } f \text{ bounds } G\}$ and $\mathscr{L}(f) = \{L(G) \mid G \text{ is an arbitrary grammar and } f \text{ bounds } G\}$.

The families $\mathscr{L}_{\mathrm{CS}}(f)$ and $\mathscr{L}(f)$ are the families of languages which we shall study. We shall consider f as a measure of derivational complexity of the languages in $\mathscr{L}_{\mathrm{CS}}(f)$ and in $\mathscr{L}(f)$.

Note that if G is an arbitrary grammar such that f bounds G, then G may have erasing rules so that $L(G)$ is not necessarily CS. However, by Lemma 2.3 we have the following result.

COROLLARY 2.9. *If f is a bounding function, then both $\mathscr{L}_{\mathrm{CS}}(f)$ and $\mathscr{L}(f)$ are families of recursive sets.*

Notation. For any function f and any constant $c > 0$, let $c \cdot f$ be the function defined for all x as

$$c \cdot f(x) = \begin{cases} 1 & \text{if } cf(x) < 1 \\ cf(x) & \text{otherwise.} \end{cases}$$

For any function f, let $\underset{\sim}{C}(f)$ be the family of functions $c \cdot f$ where $c > 0$ is computable.

From the definition of bounding functions, it is clear that if f is a bounding function, then so is every function in $\underset{\sim}{C}(f)$.

We shall be particularly concerned with the families of languages generated in linear time.

DEFINITION 2.10. A grammar G is a *linear-time* grammar if it is bounded by some bounding function which is linear. Define $\mathrm{LINEAR}_{\mathrm{CS}} = \{L(G) \mid G \text{ is a linear-time CS grammar}\}$ and $\mathrm{LINEAR} = \{L(G) \mid G \text{ is an arbitrary linear-time grammar}\}$.

We shall show in Section 5 that $\underline{\mathrm{CF}} \subsetneqq \mathrm{LINEAR}_{\mathrm{CS}} \subsetneqq \underline{\mathrm{CS}}$.

If f and g are bounding functions such that for every $x > 0$, $f(x) \leqslant g(x)$, then clearly $\mathscr{L}_{\mathrm{CS}}(f) \subseteq \mathscr{L}_{\mathrm{CS}}(g)$ and $\mathscr{L}(f) \subseteq \mathscr{L}(g)$. As a result of Theorem 3.5, we shall see that for any bounding function f and any computable $c > 0$, $\mathscr{L}_{\mathrm{CS}}(c \cdot f) = \mathscr{L}_{\mathrm{CS}}(f)$ and

$\mathscr{L}(c \cdot f) = \mathscr{L}(f)$, so that $\mathscr{L}_{\mathrm{CS}}(f) = \bigcup_{g \in \underset{\sim}{C}(f)} \mathscr{L}_{\mathrm{CS}}(g)$ and $\mathscr{L}(f) = \bigcup_{g \in \underset{\sim}{C}(f)} \mathscr{L}(g)$. Thus, we will have $\mathrm{LINEAR}_{\mathrm{CS}} = \bigcup_{g \in \underset{\sim}{C}(i)} \mathscr{L}_{\mathrm{CS}}(g) = \mathscr{L}_{\mathrm{CS}}(i)$ and $\mathrm{LINEAR} = \bigcup_{g \in \underset{\sim}{C}(i)} \mathscr{L}(g) = \mathscr{L}(i)$, where i is the identity function, $i(x) = x$. From the requirement that a bounding function be "at least linear", we shall then have $\mathrm{LINEAR}_{\mathrm{CS}} \subseteq \mathscr{L}_{\mathrm{CS}}(f)$ and $\mathrm{LINEAR} \subseteq \mathscr{L}(f)$ for every bounding function f.

While it is immediate that for every bounding function f, $\mathscr{L}_{\mathrm{CS}}(f) \subseteq \mathscr{L}(f)$, it is not known whether $\mathscr{L}(f)$ equals $\mathscr{L}_{\mathrm{CS}}(f)$. In particular, it is not known whether LINEAR equals $\mathrm{LINEAR}_{\mathrm{CS}}$.

We close this section by stating a lemma concerning restrictions on the form of the rules of grammars. This lemma is very useful in establishing positive closure properties (Section 5). It is easy to prove this result using straightforward but tedious arguments based on proofs in [16] and [17]. Recall that only e-free languages are considered.

Definition 2.11. A grammar $G = (V, \Sigma, R, X)$ is of *degree m* if

$$m = \max\{|\rho|, |\theta| \mid \rho \to \theta \in R\}.$$

Lemma 2.12. *If G is a grammar of degree $m > 2$, one can effectively construct a Type* 0 *grammar $G' = (V, \Sigma, R, X)$ such that the following conditions hold*:

(i) $L(G') = L(G)$;

(ii) *G' is* CS *if and only if G is monotonic*;

(iii) *if $\rho \to \theta \in R$, then $\rho \to \theta$ has one of the forms*

$$\left.\begin{array}{r} Z_1 \to Y_1Y_2 \\ Z_1Z_2 \to Y_1Z_2 \\ Z_1Z_2 \to Z_1Y_2 \\ Z_1Z_2 \to Z_1 \\ Z_1Z_2 \to Z_2 \\ Z_1Z_2 \to aZ_2 \\ Z_1 \to a \end{array}\right\} a \in \Sigma,\ Z_1, Z_2, Y_1, Y_2 \in V - \Sigma;$$

(iv) *for any n with $T_G(n)\downarrow$, both $T_{G'}(n)\downarrow$ and $T_{G'}(n) \leqslant MT_G(n)$ where* $M = 8m(m+1)$.

3. Connectivity in Grammars

In this section, we define the notions of a connected derivation and a connected grammar. The basic connectivity results are stated and then applied to obtain a "linear speed-up" theorem (3.5) and its corollaries. The notions of connectivity in grammars are essentially those of Gladkii [7] although some minor changes have been made in the basic definitions. Independently, Griffiths [12] has studied similar notions.

Definition 3.1. If $D : \Gamma_0 \Rightarrow \Gamma_1 \Rightarrow \cdots \Rightarrow \Gamma_n$ is a derivation in grammar $G = (V, \Sigma, R, X)$, a *production sequence* for D is a sequence of n ordered pairs, each element of the pair being an ordered triple, where the ith pair is $\langle(B_i, P_i, C_i), (B_i, Q_i, C_i)\rangle$ with $B_iP_iC_i = \Gamma_{i-1}$, $B_iQ_iC_i = \Gamma_i$, and $P_i \rightarrow Q_i \in R$.

Notice that every derivation has at least one production sequence and may have more than one. Also, notice that for each i, $0 < i < n$, $\Gamma_i = B_iQ_iC_i = B_{i+1}P_{i+1}C_{i+1}$. Thus, $\Gamma_0 = B_1P_1C_1 \Rightarrow B_1Q_1C_1 = \Gamma_1 = B_2P_2C_2 \Rightarrow B_2Q_2C_2 = \Gamma_2 = B_3P_3C_3 \Rightarrow \cdots \Rightarrow B_{n-1}Q_{n-1}C_{n-1} = \Gamma_{n-1} = B_nP_nC_n \Rightarrow B_nQ_nC_n = \Gamma_n$.

Definition 3.2. Consider a derivation $D : \Gamma_0 \Rightarrow \cdots \Rightarrow \Gamma_n$ in grammar $G = (V, \Sigma, R, X)$ and a production sequence $S = \{\langle(B_i, P_i, C_i), (B_i, Q_i, C_i)\rangle\}_{i=1}^{n}$ for D. For every $i = 0,..., n-1$,

(a) *step* $i + 1$ *occurs to the left of step* i if $|B_{i+1}P_{i+1}| \leqslant |B_i|$;

(b) *step* $i + 1$ *occurs to the right of step* i if $|P_{i+1}C_{i+1}| \leqslant |C_i|$;

(c) *step* $i + 1$ *is connected to step* i if $|B_{i+1}P_{i+1}| > |B_i|$ and $|P_{i+1}C_{i+1}| > |C_i|$.

Consider Cases (a), (b), and (c) in Definition 3.2. Since $B_{i+1}P_{i+1}C_{i+1} = B_iQ_iC_i$ and, as the left-hand side of a rewriting rule, $P_{i+1} \neq e$, it is immediate that Cases (a) and (b) cannot occur simultaneously. Now $|B_{i+1}P_{i+1}| \leqslant |B_i|$ if and only if $|C_{i+1}| \geqslant |Q_iC_i|$, and $|P_{i+1}C_{i+1}| \leqslant |C_i|$ if and only if $|B_{i+1}| \geqslant |B_iQ_i|$. Thus, the three cases are mutually exclusive and exhaust the possibilities, that is, exactly one of the following occur: (i) step $i + 1$ is connected to step i; (ii) step $i + 1$ occurs to the left of step i; (iii) step $i + 1$ occurs to the right of step i.

Definition 3.3. A derivation in a grammar is a *connected derivation* if there is a production sequence for it such that each step is connected to the previous step. A grammar is a *connected grammar* if each proper derivation is connected.

It is easy to see that if $G = (V, \Sigma, R, X)$ is a finite-state grammar such that $e \notin L(G)$, then G is a connected grammar, since in any proper derivation $D : X \Rightarrow \Gamma_1 \Rightarrow \cdots \Rightarrow \Gamma_n$, each Γ_i is in $\Sigma^+ \cup \Sigma^*V$. Note that a CF grammar cannot be connected unless it is a linear CF grammar.

In [9] it is shown that for every e-free context-free language L there is a grammar $G = (V, \Sigma, R, X)$ such that $L(G) = L$ and such that each rule in R is of one of the following forms:

$$\left.\begin{array}{l} Z \rightarrow a \\ Z \rightarrow aY_1 \\ Z \rightarrow aY_1Y_2 \end{array}\right\} a \in \Sigma,\ Z,\ Y_1,\ Y_2 \in V - \Sigma.$$

(Such a grammar is a "standard 2-form" grammar.) We can construct a connected grammar $G_1 = (V_1, \Sigma, R_1, X_1)$ such that $L(G_1) = L(G)$ and such that for any $n > 0$

both $T_G(n)\downarrow$ iff $T_{G_1}(n)\downarrow$, and $T_G(n) = T_{G_1}(n)$. The construction is as follows. Let X_1 be a new symbol and $\Sigma_1 = \{a_1 \mid a \in \Sigma\}$ be a set of new symbols, and let $V_1 = V \cup \Sigma_1 \cup \{X_1\}$. Let

$$\begin{aligned} R_1 = {} & \{X_1 \to a \mid a \in \Sigma, X \to a \in R\} \\ & \cup \{X_1 \to a_1 Y_1 \mid a \in \Sigma, Y_1 \in V - \Sigma, X \to aY_1 \in R\} \\ & \cup \{X_1 \to a_1 Y_1 Y_2 \mid a \in \Sigma, Y_1, Y_2 \in V - \Sigma, X \to aY_1Y_2 \in R\} \\ & \cup \{b_1 Z \to ba_1, b_1 Z \to ba \mid b, a \in \Sigma, Z \in V - \Sigma, Z \to a \in R\} \\ & \cup \{b_1 Z \to ba_1 Y_1 \mid b, a \in \Sigma, Z, Y_1 \in V - \Sigma, Z \to aY_1 \in R\} \\ & \cup \{b_1 Z \to ba_1 Y_1 Y_2 \mid b, a \in \Sigma, Z, Y_1, Y_2 \in V - \Sigma, Z \to aY_1Y_2 \in R\}. \end{aligned}$$

In the above construction it is obvious that $L(G_1) = L(G)$. The symbols of Σ_1 play the role of "connectors," since by the form of the rules if $X_1 \Rightarrow \Gamma_1 \Rightarrow \cdots \Rightarrow \Gamma_n$ is a proper derivation in G_1, then $i < n$ implies that Γ_i has exactly one symbol from Σ_1 and Γ_n has one symbol from Σ_1 or is a string from which nothing can be derived. However, the preservation of the time function stems from a property that is unique to CF grammars: any proper derivation of a terminal string can be transformed into a proper derivation of that string such that at each step the leftmost nonterminal symbol is the symbol transformed at that step.

It is easy to see that for an arbitrary grammar G one can construct a connected grammar G' such that $L(G') = L(G)$. This follows from the fact that for any grammar G one can construct a nondeterministic one-tape off-line Turing machine that "simulates" G. Since a Turing machine computes by scanning adjacent squares in successive moves, the grammar obtained from this machine will be connected. However, the "obvious" construction—which begins with grammar G, produces Turing machine M which simulates G, and then produces a connected grammar G' from M—may not preserve the order of the time function of G; in fact, it is not unusual for $T_{G'}$ to be on the order of $(T_G)^2$ (the simulation of grammars by Turing machines is discussed further in Section 4). What is needed here is a construction of a connected grammar G' from G such that $L(G') = L(G)$ and $T_{G'} \in \underset{\sim}{C}(T_G)$.

Gladkii [7] has established the following:

If G is a grammar, one can construct a connected grammar G' and find a constant k (which depends only on G) such that $L(G') = L(G)$ and such that for any $n > 0$, if $T_G(n)\downarrow$, then $T_{G'}(n)\downarrow$, and $T_{G'}(n) \leqslant kT_G(n)$.

We shall use a variation of this result which is stated below. A detailed proof appears in [1].

Connectivity Theorem 3.4. *If G is a grammar, one can effectively construct a connected Type* 0 *grammar $G_1 = (V, \Sigma, R, X)$ and a constant $k > 0$ such that*

(i) $L(G_1) = L(G)$;
(ii) G_1 *is* CS *if and only if G is monotonic*;

(iii) *every proper connected derivation in G_1 has a unique production sequence*;
(iv) *for any $n > 0$, if $T_G(n)\downarrow$, then both $T_{G_1}(n)\downarrow$ and $T_{G_1}(n) \leqslant kT_G(n)$*;
(v) *if f is a bounding function such that f bounds G, then function $k \cdot f$ bounds G_1.*

Theorem 3.4 is a powerful tool for studying time-bounded grammars and the languages they generate. The first application of this result is a "linear speed-up" theorem for grammars similar to that of Hartmanis and Stearns [13] for multitape Turing machines.

THEOREM 3.5. *Let G be a grammar and let f be a bounding function that bounds G. For any positive computable number c, one can effectively construct a connected grammar G_0 such that $L(G_0) = L(G)$ and the function $c \cdot f$ bounds G_0.*

Proof. There is nothing to prove if $c \geqslant 1$ so assume $c < 1$. Let m be the least positive integer such that $2/m < c$. Since c is computable, m can be found effectively.

By Theorem 3.4 there is no loss of generality if one assumes that $G = (V, \Sigma, R, X)$ is connected and that each proper connected derivation in G has a unique production sequence. Let $k = \max\{|\theta| - |\rho| \mid \rho \to \theta \in R\}$. Thus, if $\Gamma_0 \Rightarrow \cdots \Rightarrow \Gamma_t$ in G, then $|\Gamma_t| - |\Gamma_0| \leqslant kt$, since length can be increased by at most k at each step.

Construct grammar G_0 as follows:

(i) Let X_0 be a new symbol and let $R_{\text{SHORT}} = \{X_0 \to w \mid w \in \bigcup_{i \leqslant (2m+1)k} L(G, V^i)\}$. Since f is recursive and f bounds G, $\bigcup_{i \leqslant (2m+1)k} L(G, V^i)$ can be effectively found so that R_{SHORT} can be effectively constructed.

(ii) For each $i = 1,..., (2m + 1)k$ and each $j = 1,..., m$ let $R_{i,j} = \{\Gamma \to \psi \mid \Gamma, \psi \in V^*, |\Gamma| \leqslant i$, there exist $\theta_0,..., \theta_j \in V^*$ such that $\theta_0 = \Gamma$, $\theta_j = \psi$, and $\theta_0 \Rightarrow \cdots \Rightarrow \theta_j$ is a connected derivation in $G\}$. Since $|\Gamma| \leqslant i \leqslant (2m + 1)k$, each $R_{i,j}$ can be effectively constructed.

(iii) Let $R_0 = R \cup R_{\text{SHORT}} \cup (\bigcup_{i=1}^{(2m+1)k} \bigcup_{j=1}^{m} R_{i,j})$, let $V_0 = V \cup \{X_0\}$, and let $G_0 = (V_0, \Sigma, R_0, X_0)$.

We must show that (A) $L(G_0) = L(G)$, (B) G_0 is connected, and (C) $c \cdot f$ bounds G_0.

(A) From the construction of R_0, it is immediate that for any $w \in V^+$, $X_0 \overset{+}{\Rightarrow} w$ in G_0 only if $X \overset{+}{\Rightarrow} w$ in G, so that $L(G_0, V^+) \subseteq L(G, V^+)$. But $R \subseteq R_0$; so $L(G, V^+) \subseteq L(G_0, V^+)$. The symbol X_0 can only occur once in any proper derivation in G_0 and in that occurrence serves as the initial symbol. Hence, $L(G_0, V_0^+) = L(G_0, V^+) = L(G, V^+)$ and $L(G_0) = L(G)$.

(B) Let $D : X_0 \Rightarrow \theta_1 \Rightarrow \cdots \Rightarrow \theta_n$ be a proper derivation in G_0. By the construction of R_0, for each $i = 1,..., m$, either $\theta_{i-1} \Rightarrow \theta_i$ in G or there exist $\pi_{i,1},..., \pi_{i,\rho(i)} \in V^+$ such that $\theta_{i-1} \Rightarrow \pi_{i,1} \Rightarrow \cdots \Rightarrow \pi_{i,\rho(i)} \Rightarrow \theta_i$ is a connected derivation in G, where $\theta_0 = X$. Thus $D' : X \Rightarrow \cdots \Rightarrow \theta_1 \Rightarrow \cdots \Rightarrow \theta_n$ is a proper derivation in G.

Since G is connected, D' is a connected derivation and so D is a connected derivation in G_0. Thus G_0 is a connected grammar.

(C) Suppose $D : X \Rightarrow \theta_1 \Rightarrow \cdots \Rightarrow \theta_\sigma$ is a derivation in G and $|\theta_\sigma| < (2m+1)k$. Then $X_0 \to \theta_\sigma$ is a rule in $R_{\mathrm{SHORT}} \subseteq R_0$, so that $t_{G_0}(\theta_\sigma) = 1 \leqslant c \cdot f(|\theta_\sigma|)$. Thus if $T_{G_0}(n)\downarrow$ and $n < (2m+1)k$, then $T_{G_0}(n) = 1 \leqslant c \cdot f(n)$.

Suppose $D : X \Rightarrow \theta_1 \Rightarrow \cdots \Rightarrow \theta_\sigma$ is a proper connected derivation in G such that $|\theta_\sigma| \geqslant (2m+1)k$ and that $\sigma = t_G(\theta_\sigma)$. By choice of k, $t_G(\theta_\sigma) \geqslant |\theta_\sigma|/k$ so that $\sigma \geqslant |\theta_\sigma|/k \geqslant 2m+1$. Let $S = \{\langle(B_i, P_i, C_i), (B_i, Q_i, C_i)\rangle\}_{i=1}^{\sigma}$ be a production sequence for D such that each step is connected to the previous step. Let ρ be the greatest integer such that $\rho m \leqslant \sigma$. By the construction of R_0,

$$D' : X_0 \Rightarrow \theta_m \Rightarrow \cdots \Rightarrow \theta_{\rho m}$$

is a proper connected derivation in G_0.

There are two cases:

(i) If $\rho m = \sigma$, then $\theta_m = \theta_\sigma$ and $t_{G_0}(\theta_\sigma) \leqslant \rho = \sigma/m < 2\sigma/m \leqslant c\sigma = ct_G(\theta_\sigma) \leqslant cT_G(|\theta_\sigma|) \leqslant cf(|\theta_\sigma|)$.

(ii) If $\rho m < \sigma$, then $\theta_{\rho m} \Rightarrow \cdots \Rightarrow \theta_\sigma$ is a connected derivation in G of length $\sigma - \rho m < m$. Consider the production sequence $\{\langle(B_i, P_i, C_i), (B_i, Q_i, C_i)\rangle\}_{i=m+1}^{\sigma}$. Then there exist B'' and C'' such that each $B_iP_iC_i$ can be expressed as $B''\alpha_iP_iB_iC''$ and each $B_iQ_iC_i$ can be expressed as $B''\alpha_iQ_i\beta_iC''$ where $|\alpha_i| \leqslant (\sigma - \rho m)\, k < mk$ and $|\beta_i| \leqslant (\sigma - \rho m)\, k < mk$ (recall that the derivation is connected and $k = \max\{|\rho|, |\theta| \mid \rho \to \theta \in R\}$). Thus $\alpha_{\rho m+1}P_{\rho m+1}\beta_{\rho m+1} \Rightarrow \cdots \Rightarrow \sigma_\sigma Q_\sigma \beta_\sigma$ is a connected derivation in G with $|\alpha_{\rho m+1}P_{\rho m+1}\beta_{\rho m+1}| < (2m+1)k$; so $\alpha_{\rho m+1}P_{\rho m+1}\beta_{\rho m+1} \to \alpha_\sigma Q_\sigma \beta_\sigma$ is a rule in R_0. Hence $\theta_{\rho m} = B_{\rho m+1}P_{\rho m+1}C_{\rho m+1} = B''\alpha_{\rho m+1}P_{\rho m+1}\beta_{\rho m+1}C''$ and $\theta_\sigma = B''\alpha_\sigma Q_\sigma \beta_\sigma C''$, so that $X_0 \Rightarrow \theta_m \Rightarrow \cdots \Rightarrow \theta_{\rho m} \Rightarrow \theta_\sigma$ is a derivation of length $\rho + 1$ in G_0. Thus $t_{G_0}(\theta_\sigma) < \rho + 1 = \sigma/m + 1 \leqslant 2\sigma/m \leqslant c\sigma = ct_G(\theta_\sigma) \leqslant cT_G(|\theta_\sigma|) \leqslant cf(|\theta_\sigma|)$.

Thus for any n such that $T_{G_0}(n)\downarrow$, $T_{G_0}(n) \leqslant c \cdot f(n)$; so $c \cdot f$ bounds G_0.

By (A), (B), and (C), G_0 has the desired properties.

COROLLARY 3.6. *Let G be a grammar and f a bounding function such that f bounds G. For any positive computable number c, one can construct a connected grammar G' and a positive integer k such that $L(G') = L(G)$, and whenever $T_{G'}(n)\downarrow$, $T_{G'}(n) = 1$ if $n < k$ and $T_{G'}(n) \leqslant cf(n)$ if $n \geqslant k$.*

COROLLARY 3.7. *If f is a bounding function and c is a positive computable number, then $\mathscr{L}_{\mathrm{CS}}(f) = \mathscr{L}_{\mathrm{CS}}(c \cdot f)$ and $\mathscr{L}(f) = \mathscr{L}(c \cdot f)$. Thus, $\mathscr{L}_{\mathrm{CS}}(f) = \bigcup_{g \in \underline{C}(f)} \mathscr{L}_{\mathrm{CS}}(g)$ and $\mathscr{L}(f) = \bigcup_{g \in \underline{C}(f)} \mathscr{L}(g)$.*

COROLLARY 3.8. *If i is the function by $i(x) = x$ for all x, then* LINEAR$_{\mathrm{CS}}$ $= \mathscr{L}_{\mathrm{CS}}(i)$ *and* LINEAR $= \mathscr{L}(i)$.

COROLLARY 3.9. *For any bounding function* f, $\text{LINEAR}_{\text{CS}} \subsetneq \mathscr{L}_{\text{CS}}(f)$ *and* $\text{LINEAR}_{\text{CS}} \subsetneq \text{LINEAR} \subsetneq \mathscr{L}(f)$.

Proof. Since f is a bounding function, there is a constant $k > 0$ such that for all n, $kn \leqslant f(n)$. Thus $\text{LINEAR}_{\text{CS}} = \mathscr{L}_{\text{CS}}(i) = \mathscr{L}_{\text{CS}}(k \cdot i) \subsetneq \mathscr{L}_{\text{CS}}(f)$ and $\text{LINEAR}_{\text{CS}} \subsetneq \text{LINEAR} = \mathscr{L}(i) = \mathscr{L}(k \cdot i) \subsetneq \mathscr{L}(f)$.

COROLLARY 3.10. *If f is a bounding function and G is a grammar such that f bounds G, then one can effectively find a grammar G' and a constant $c > 0$ such that $L(G') = L(G)$ and such that for any n where $T_{G'}(n)\downarrow$, $T_{G'}(n) \leqslant cf(n)$.*

Proof. Let G' and k be as in Corollary 3.6. Consider those n such that $T_{G'}(n)\downarrow$. If $f(1) \geqslant 1$, then for $n < k$, $T_{G'}(n) = 1 \leqslant f(1) \leqslant f(n)$ since f is nondecreasing. Since $f(n) \geqslant T_{G'}(n)$ for $n \geqslant k$, $c = 1$ will do. If $f(1) < 1$, then for $n < k$, $T_{G'}(n) = 1 < (1/f(1))^2 f(1) \leqslant (1/f(1))^2 f(n)$ since f is nondecreasing. Since $1/f(1) > 1$, for $n \geqslant k$, $T_{G'}(n) \leqslant f(n) \leqslant (1/f(1))^2 f(n)$. Hence $c = (1/f(1))^2$ will do.

We now turn to an application of connectivity to a certain class of linear-time grammars. Suppose $G = (V, \Sigma, R, X)$ is a CS grammar such that each rule in R is of one of the following forms:

$$\left.\begin{array}{r}
Z_1 \to Y_1 \\
Z_1 \to Y_1Y_2 \\
Z_1Z_2 \to Y_1Z_2 \\
Z_1Z_2 \to Z_1Y_2 \\
Z_1Z_2 \to aZ_2 \\
aZ_2 \to aZ_3 \\
Z_1 \to a
\end{array}\right\} a \in \Sigma,\ Z_1, Z_2, Y_1, Y_2 \in V - \Sigma.$$

We lose no generality assuming that G is connected; hence, every proper derivation of G is connected.

Let L be the set of those $w \in L(G)$ such that there is a connected proper derivation $X \Rightarrow \phi_1 \Rightarrow \cdots \Rightarrow \phi_n = w$ in G with the following property: for some $k \leqslant n$, every rule used in the subderivation $X \Rightarrow \phi_1 \Rightarrow \cdots \Rightarrow \phi_k$ is a CF rule and every rule used in the subderivation $\phi_k \Rightarrow \cdots \Rightarrow \phi_n = w$ is a length-preserving rule. We shall show that if there is an integer $p > 0$ such that every such derivation has length bounded by $p \mid w \mid$, then L is CF.

Let L_1 be the set of all strings $\bar{w} \in V^*$ such that there is a proper derivation of $\bar{w}$ in G which uses only CF rules. Hence, L_1 is a CF language over V. We view L_1 as the set of strings obtained by the "CF-subderivation" of the type described above. To any CF grammar G_1 such that $L(G_1) = L_1$, add the set of rules $\{Z \to (Z, a) \mid a \in \Sigma, Z \in V\}$. Let the resulting language be L_2. Since this set of rules is CF and G_1 is a CF grammar, L_2 is CF.

Consider a nondeterministic one-tape off-line Turing acceptor M which operates as follows. Upon receiving input over $V \times \Sigma$, say $w_1 = (Z_1, a_1) \cdots (Z_n, a_n)$, M imitates some derivation $Z_1 \cdots Z_n \Rightarrow \psi_1 \Rightarrow \cdots \Rightarrow \psi_t$ in G which is connected and which uses only length-preserving rules. If $\psi_t = a_1 \cdots a_n$, then M accepts w_1.

Since the derivations have length bounded by $f(x) = px$ and since the imitated derivations of G are connected, M operates within a linear time bound. In [14], Hennie shows that a deterministic one-tape off-line Turing acceptor which operates in linear time accepts only a regular set. This result may be extended to nondeterministic machines in a straightforward manner. Hence, if L_3 is the language accepted by M, then L_3 is regular.

Let $h : (V \times \Sigma)^* \to \Sigma^*$ be the homomorphism determined by defining $h((Z, a)) = a$ for $Z \in V$, $a \in \Sigma$. Let $L_4 = h(L_2 \cap L_3)$. Since L_2 is CF and L_3 is regular, and since the family of CF languages is closed under homomorphic mappings, L_4 is CF. But clearly, $L_4 = L$ so that L is CF.

This argument verifies one's intuitive feelings that a language like

$$L = \{a^n b^n c^n \mid n \geqslant 1\}$$

cannot be generated in linear time by first generating a string in $a^*b^*c^*$ and then checking to see that the number of a's matches the number of b's, etc. This argument does not show that L is not in $\text{LINEAR}_{\text{CS}}$. In fact, it is shown in Section 7 that $\text{LINEAR}_{\text{CS}}$ contains every language accepted in linear time by a nondeterministic multicounter acceptor, and hence contains L.

4. Simulation of Grammars by Machines

It is natural to ask what relationship exists between the generation of a language by a grammar within a given time bound and the recognition of that language by a Turing acceptor. In particular, if f is a bounding function and G is a grammar such that f bounds G, does there exist a nondeterministic Turing machine M (multitape, multihead, or whatever) such that M recognizes $L(G)$ within time bound g for some $g \in \underset{\sim}{C}(f)$?

One approach to answering this question is to determine a machine model in which we can simulate derivations of grammars of some given family. For example, if G is a Type 3 grammar, then one can construct a nondeterministic finite state machine M such that M accepts $L(G)$ in real time[4] and such that the computations of M essentially imitate the proper derivations of G. Similarly, if G is a standard 2-form context-free grammar [9], then one can construct a nondeterministic pushdown store machine M such that M accepts $L(G)$, M operates in real time, and the computations of M essentially imitate the proper left-to-right derivations of G (see [8]). In the first case,

[4] That is, if M accepts an input w, it does so by means of a computation of $| w |$ steps.

the time function T_G is $T_G(n) = n$. In the second, if $T_G(n)\downarrow$, then $T_G(n) \leqslant n$. Hence M preserves the order of the time bound.

Since for any grammar $G = (V, \Sigma, R, X)$, $L(G, V^+)$ is a recursively enumerable set, one can construct a one-tape off-line deterministic Turing machine M with the property that if $\theta_1 \Rightarrow \cdots \Rightarrow \theta_n$ is a derivation in G, then there is a computation $\pi_0 \vdash \cdots \vdash \pi_m$ of M and a sequence of integers $i_1, \ldots, i_n$ such that for any $j = 1, \ldots, n$, the tape content of π_{i_j} is an encoding of θ_j. However, the length m of the computation $\pi_0 \vdash \cdots \vdash \pi_m$ is generally of an order of at least n^2. In fact, if G is bounded by f, then there is a one-tape off-line nondeterministic Turing machine M which imitates the derivation of G within time $g \in \underset{\sim}{C}(f^2)$, but not necessarily a machine M' of this type which imitates the derivations of G within time bound h where $\lim_{n\to\infty} h(n)/(f(n))^2 = 0$.

There are two basic reasons why the time function of an "imitating" off-line machine may reach the sequare of the time function of the grammar:

(1) In any arbitrary derivation $D : \theta_0 \Rightarrow \cdots \Rightarrow \theta_n$ of an arbitrary grammar G, for any i, $0 < i < n$, the step $\theta_i \Rightarrow \theta_{i+1}$ may occur arbitrarily far to the right or to the left of the step $\theta_{i+1} \Rightarrow \theta_i$. But the simulation by machine must take place as if D were connected, since the process of reading and writing on a Turing tape is a connected process—i.e., adjacent tape squares are scanned in successive moves. Thus, if G is not connected, particularly if D is not connected, then as many as $|\theta_{i-1}|$ steps must be added to the simulation process.

(2) If $\alpha\pi\beta \Rightarrow \alpha\theta\beta$ in G where $|\theta| > |\pi|$, then on the corresponding simulation machine tape we must either move α to the left or move β to the right in order to make room for the $|\theta| - |\pi|$ extra symbols, and this takes a number of machine steps which is on the order of $|\alpha|$ or $|\beta|$. A similar process takes place if $|\theta| < |\pi|$. This process may occur in a grammar to an arbitrary depth of embedding and arbitrarily far from either end of the tape. Thus the simulation process may need on the order of m^2 extra steps, where m is the maximum length of any string generated.

One method of overcoming (1) is to restrict our attention to connected grammars, so that each proper derivation is connected. By Theorem 3.4 we lose no generality if our interest is in the recognition of languages by means of imitating derivations.

We cannot overcome (2) on a one-tape off-line machine since it is known that a one-tape off-line nondeterministic Turing acceptor recognizes only regular sets in linear time [14], and as indicated above (and shown in Section 5), for every context-free language L there is a grammar G such that $L(G) = L$ and for all n such that $T_G(n)\downarrow$, $T_G(n) \leqslant n$. Thus, we turn to on-line multitape Turing acceptors (with one head per tape). We consider machines with two pushdown store working tapes and one input tape. We visualize the two pushdown tapes as being "head-to-head," the first "pushing down to the left, the second "pushing down" to the right in the following way.

Let $G = (V, \Sigma, R, X)$ be a grammar and $Z_1 \cdots Z_m \to Y_1 \cdots Y_n \in R$, each Z_i, $Y_i \in V$. For any strings $A_1 \cdots A_t$, $B_1 \cdots B_s$ such that each A_i, $B_i \in V$, we wish to imitate the derivation $A_1 \cdots A_t Z_1 \cdots Z_m B_1 \cdots B_s \Rightarrow A_1 \cdots A_t Y_1 \cdots Y_n B_1 \cdots B_s$. Suppose that for some j, $1 \leqslant j \leqslant m$, $A_1 \cdots A_t Z_1 \cdots Z_j$ is stored on pushdown Tape 1 with the read-write head scanning Z_j, and that $Z_{j+1} \cdots Z_m B_1 \cdots B_s$ is stored in pushdown Tape 2 with the read-write head scanning Z_{j+1}.

The machine operates by first reading $Z_{j+1} \cdots Z_{m-1}$ off pushdown Tape 2 until Z_m is scanned, storing $Z_{j+1} \cdots Z_{m-1}$ on pushdown Tape 1 so that $A_1 \cdots A_t Z_1 \cdots Z_{m-1}$ is stored on Tape 1 with Z_{m-1} being scanned. Then $Z_2 \cdots Z_{m-1}$ is read off pushdown Tape 1 and stored on pushdown Tape 2, so that Tape 1 contains $A_1 \cdots A_t Z_1$ with Z_1 being scanned, and Tape 2 contains $Z_2 \cdots Z_m B_1 \cdots B_s$ with Z_2 being scanned. This action essentially checks to see that $Z_1 \cdots Z_m$ occurs so that the rule $Z_1 \cdots Z_m \rightarrow Y_1 \cdots Y_n$ can be applied. At this point, $Y_1 \cdots Y_n$ is written on Tape 1 in place of Z_1, and $Z_2 \cdots Z_m$ is erased from Tape 2. Finally, for some k, $1 \leqslant k \leqslant n$, $Y_{k+1} \cdots Y_n$ is read off Tape 1 and stored on Tape 2.

Thus, if G is a connected grammar, any step in a derivation of G can be imitated by at most $4M$ steps by a two pushdown store machine, where M is the degree of G.

From the above discussion, the following is immediate:

LEMMA 4.1. *Let $G = (V, \Sigma, R, X)$ be a connected grammar and let f be a bounding function that bounds G. One can effectively construct a nondeterministic Turing machine M with two pushdown store tapes (and no other tapes) and an integer $k > 0$ such that:*

(i) *for every $w \in L(G, V^+)$, there is a computation of M of length no greater than $kf(|w|)$ which begins with tape content X and halts with tape content w;*

(ii) *every computation of M begins with tape content X, and if it halts with tape content w, then $w \in L(G, V^+)$.*

We now have our theorem on "simulation."

THEOREM 4.2. *If $G = (V, \Sigma, R, X)$ is a grammar and f is a bounding function such that f bounds G, then one can effectively construct an on-line nondeterministic Turing acceptor M with three pushdown store tapes such that $L(M) = L(G)$*[5] *and such that for each $w \in L(G)$ there is a computation of M of length no greater than* $\max(f(|w|), |w|)$ *which accepts w.*

Proof. By Theorem 3.4, we may assume that G is connected. Using the methods of [13], we can modify the machine M_1 of Lemma 4.1 to construct a machine M_2 with two pushdown store tapes such that for any proper derivation $X \Rightarrow \theta_1 \Rightarrow \cdots \Rightarrow \theta_n$ in G, there is a computation of M_1 of length no greater than $n/4$ which halts with an encoding of θ_n of length no more than $|\theta_n|/4$ on the two tapes.

From M_2 we construct a machine M_3 by adding an input tape, which is read only from left to right, and an additional pushdown store tape which we shall call Tape 3.

[5] $L(M)$ is the set of inputs accepted by M.

The two pushdown store tapes from M_2 will be called Tapes 1 and 2. M_3 operates in three phases.

Phase 1

In Phase 1, M_3 operates like M_2, imitating an arbitrary proper derivation of G on Tapes 1 and 2. Simultaneously, the input w is read, one symbol per step, and stored on Tape 3, compressed (as in [13]) by a factor of four so that no more than $|w|/4$ tape squares are needed.

This phase ends when M_3 guesses that the contents of Tapes 1 and 2 are an encoding of a string $w' \in L(G)$.[6]

Phase 2

In Phase 2, M_3 positions the tape contents of Tapes 1 and 2 so that Tape 1 holds the encoding of a prefix of w', say w_1, with the tape head at the right end of the encoding of w_1, and Tape 2 holds the encoding of a suffix of w', say w_2, with the tape head at the left end of the encoding of w_2, where $w' = w_1 w_2$. The particular choice of w_1 and w_2 is made by guessing that at the end of Phase 2, the input read and stored on Tape 3 has length equal to $|w_1|$.

Meanwhile, M_3 continues to read input if the input has not already been completely stored on Tape 3. This phase ends when the positioning of Tapes 1 and 2 is completed.

Phase 3

In Phase 3, M_3 compares the contents of Tape 1 with the input read up to this point and simultaneously compares the contents of Tape 2 with any additional input still to be read. If $w_1 w_2 = w$, then M_3 halts in an accepting state. Otherwise, M_3 continues to operate in some arbitrary nonaccepting manner.

Thus M_3 accepts w if and only if $w \in L(G)$. Further, for any $w \in L(G)$, there is a proper derivation of w in G of length no greater than $f(|w|)$. This derivation can be imitated in Phase 1 by a computation of no more than $f(|w|)/4$ steps. Since w is stored in only $|w|/4$ tape squares, Phase 2 takes at most $|w|/4$ steps. Finally, Phase 3 takes $|w|/4$ steps if the input is entirely stored on Tape 3 by the end of Phase 2 and takes $\max(|w|/4, |w| - (f(|w|)/4 + |w|/4))$ steps otherwise. Thus this minimum length accepting computation takes at most

$$f(|w|)/4 + |w|/4 + \max(|w|/4, |w| - (f(|w|)/4 + |w|/4)) \leqslant \max(|w|, f(|w|))$$

steps.

Theorem 4.2 can be strengthened if one looks at different classes of bounding functions. If $\lim_{n\to\infty} f(n)/n \geqslant 3$, the Turing machine need have only two pushdown store tapes if one allows it to operate by first imitating a derivation on these tapes and

[6] The ability of a nondeterministic machine to "guess" is described in [2].

then comparing the result with the input, so the input is not read until the end of the computation. Since this type of comparison takes no more than $2n$ steps, a speed-up of another $1/3$ allows one to obtain the same result with respect to preservation of time. If $\lim_{n\to\infty} f(n)/n^2 > 0$, the input can be read first, stored on the "bottom" of one of the pushdown store tapes. After imitating some derivation on the pushdown store tapes, this stored input can be compared with the result of the imitated derivation. This comparison need take no more than kn^2 steps for some $k > 0$; so again time can be preserved.

Note that the converse of Theorem 4.2 does not hold for linear functions. It is easy to see that the language $\{wcw^Rcw \mid w \in \{a, b\}^*\}$ can be accepted in real time by a deterministic on-line Turing acceptor with one storage tape. However, in Section 6 it is shown that the language cannot be generated by a grammar G such that $\lim_{n\to\infty} T_G(n)/n^2 = 0$.

It is well known that the family of CS languages is precisely the family of languages accepted by nondeterministic linear-bounded automata. In [17] Landweber showed that if M is a deterministic linear-bounded automaton, then $L(M)$ is CS. In [16] Kuroda showed that if M is a nondeterministic linear-bounded automaton, then $L(M)$ is CS, and that if G is a CS grammar, then $L(G)$ is accepted by some nondeterministic linear-bounded automaton. The arguments in [16] and [17] can be extended to time-bounded models to yield the following propositions:

PROPOSITION 4.3. *If M is a one tape off-line nondeterministic Turing acceptor which accepts within time bound f (where f is a bounding function), then one can construct a connected grammar G such that $L(G) = L(M)$ and such that f bounds G. In this case, G essentially imitates the action of M.*

PROPOSITION 4.4. *Let f be a bounding function such that for some $c \geqslant 1$ and all x, $f(x) \geqslant cx^2$. A language L is accepted by a nondeterministic linear-bounded automaton that accepts within time bound f if and only if $L \in \mathscr{L}_{CS}(f)$.*

Similarly, we have:

PROPOSITION 4.5. *Let f be a bounding function such that for some $c \geqslant 1$ and all x, $f(x) \geqslant cx^2$. A language L is accepted by a nondeterministic one-tape off-line Turing acceptor which accepts within time bound f if and only if $L \in \mathscr{L}(f)$.*

5. POSITIVE CLOSURE AND CONTAINMENT PROPERTIES

In this section positive closure and containment properties of the families $\mathscr{L}_{CS}(f)$ and $\mathscr{L}(f)$ are studied. First, it is shown that LINEAR$_{CS}$ (hence, $\mathscr{L}_{CS}(f)$ and $\mathscr{L}(f)$) contains the e-free CF languages. Then, it is shown that LINEAR$_{CS}$ contains CS languages which are not CF by showing that $\mathrm{LINEAR}_{\mathrm{CS}}$ forms a basis for the recursively enumerable sets.

LEMMA 5.1. *If L is an e-free* CF *language, then* $L \in \mathrm{LINEAR}_{\mathrm{CS}}$.

Proof. By the results of [9], there is a CF grammar $G = (V, \Sigma, R, X)$ such that $L(G) = L$ and such that each rule in R is of one of the forms

$$\left.\begin{array}{l} Z \to a \\ Z \to aY_1 \\ Z \to aY_1Y_2 \end{array}\right\} Z,\ Y_1,\ Y_2 \in V - \Sigma,\ a \in \Sigma.$$

Since each rule in R either is length-increasing or transforms a nonterminal symbol into a terminal symbol, it is clear that for any n such that $T_G(n){\downarrow}$, $T_G(n) \leqslant n$. Thus G is bounded by $i(n) = n$, so $L(G) \in \mathscr{L}_{\mathrm{CS}}(i) = \mathrm{LINEAR}_{\mathrm{CS}}$.

LEMMA 5.2. *For any finite set Σ and any recursively enumerable set $L \subseteq \Sigma^*$, there is a language $L' \in \mathrm{LINEAR}_{\mathrm{CS}}$ and a homomorphism h such that $h[L'] = L$.*

Proof. Let $G = (V, \Sigma, R, X)$ be a Type 0 grammar such that $L(G) = L$. We lose no generality by assuming that each rule in R is of one of the forms

$$\left.\begin{array}{l} \rho \to \theta \\ Z \to a \end{array}\right\} \rho,\ \theta \in (V - \Sigma)^*,\ Z \in V - \Sigma,\ a \in \Sigma.$$

Let X', D, and d be three new symbols, and let $V' = V \cup \{X', D, d\}$ and $\Sigma_d = \Sigma \cup \{d\}$. Let

$$\begin{aligned} R' = {} & \{X' \to dX,\ dD \to dd\} \\ & \cup \{\rho \to \theta \mid \rho, \theta \in (V - \Sigma)^*, |\rho| < |\theta|, \rho \to \theta \in R\} \\ & \cup \{\rho \to D^k\theta \mid \rho, \theta \in (V - \Sigma)^*, |\rho| \geqslant |\theta|, \rho \to \theta \in R, k = |\rho| - |\theta| + 1\} \\ & \cup \{Z \to a \mid Z \in V - \Sigma, a \in \Sigma, Z \to a \in R\} \\ & \cup \{ZD \to DDZ \mid Z \in V - \Sigma\}. \end{aligned}$$

If $G' = (V', \Sigma_d, R', X')$ then $L(G') \subseteq \{d\}^+ L(G)$ and for each $w \in L(G)$, there is an integer $k > 0$ such that $d^k w \in L(G')$. Let $h : \Sigma_d{}^* \to \Sigma^*$ be the homomorphism determined by defining $h(d) = e$ and for $a \in \Sigma$, $h(a) = a$. Then $h[L(G')] = L(G) = L$. Further, since each rule in R' is either a length-increasing rule or of the forms $dD \to dd$ or $Z \to a$, it is immediate that for any n such that $T_{G'}(n){\downarrow}$, $T_{G'}(n) \leqslant 2n$. Thus $L(G') \in \mathrm{LINEAR}_{\mathrm{CS}}$.

THEOREM 5.3. *For any bounding function f, $\mathscr{L}_{\mathrm{CS}}(f)$ and $\mathscr{L}(f)$ contain the e-free* CF *languages and also* CS *languages which are not* CF.

Proof. Consider the languages L and L' in Lemma 5.2. If L is not CF, then neither is L', since the CF languages are closed under arbitrary homomorphic mappings. Since L' is in $\text{LINEAR}_{\text{CS}}$, it is CS.

Gladkii [7] showed that $\text{LINEAR}_{\text{CS}}$ contains the e-free CF languages and also contains non-CF languages. The proofs here are different from those in [7].

We now study the positive closure properties of the families $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$. For the most part we investigate AFL [6] operations, but we also explore reversal,[7] intersection, and substitution.

It is a straightforward exercise to show that the families $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under union, concatenation, and reversal, the arguments being based on simple grammar constructions. Similarly, the "cross-product" construction used in [4] to show that the CF languages are closed under intersection with regular sets can be modified to show that the families $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under intersection with regular sets. Thus we state the following theorem without proof.

THEOREM 5.4. *For any bounding function f, the families $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under union, concatenation, reversal, and intersection with regular sets.*

We now investigate the closure of $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ under certain mapping properties. We begin by showing closure under $\text{LINEAR}_{\text{CS}}$-substitution.[8] Recall that for any f, each language in $\mathscr{L}_{\text{CS}}(f)$ or $\mathscr{L}(f)$ is e-free. Hence every $\mathscr{L}_{\text{CS}}(f)$-substitution and every $\mathscr{L}(f)$-substitution is e-free.

THEOREM 5.5. *If f is a bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under* $\text{LINEAR}_{\text{CS}}$-substitution.

Proof. This proof involves a straightforward but tedious argument, and so only a sketch will be given (the details may be found in [1]).

Let $G = (V, \Sigma, R, X)$ be a grammar such that f bounds G. Assume that if $\rho \to \theta \in R$, then $\rho \in (V - \Sigma)^*$. Let $c > 0$ be a constant such that for all $n \geqslant 0$, $T_G(n)\downarrow$ implies $T_G(n) \leqslant cf(n)$.

Let τ be a $\text{LINEAR}_{\text{CS}}$-substitution on Σ. For each $a \in \Sigma$, let $G_a = (V_a, \Sigma_a, R_a, X_a)$ be a CS grammar such that G_a is linear-time and such that $L(G_a) = \tau(a)$. Assume that if $\rho \to \theta \in R_a$, then $\rho \in (V_a - \Sigma_a)^*$. Let $c_a > 0$ be a constant such that $T_{G_a}(a)\downarrow$

[7] If $w \in \Sigma^*$, the *reversal* of w, w^R is defined as follows: if $w = e$, then $w^R = e$; if $w = a \in \Sigma$, then $w^R = a$; if $w = a_1 \cdots a_n$, $n \geqslant 1$, $a_i \in \Sigma$, then $w^R = a_n \cdots a_1$. If $L \subseteq \Sigma^*$, the *reversal* of L is $L^R = \{w^R \mid w \in L\}$.

[8] Let Σ be a finite nonempty set of symbols. For each $a \in \Sigma$, let Σ_a a finite nonempty set of symbols, and let $\tau(a)$ be a subset of Σ_a^*. Let $\tau(e) = \{e\}$, $\tau(a_1 \cdots a_n) = \tau(a_1) \cdots \tau(a_n)$ for $n \geqslant 1$, $a_i \in \Sigma$, and $\tau[L] = \bigcup_{w \in L} \tau(w)$ for $L \subseteq \Sigma^*$. Then τ is a *substitution on* Σ, and is an *e-free* substitution if for each $a \in \Sigma$, $e \notin \tau(a)$. If $\mathscr{F}$ is a family of languages and $\tau(a) \in \mathscr{F}$ for each $a \in \Sigma$, then τ is an *$\mathscr{F}$-substitution.*

implies $T_{G_a}(n) \leqslant c_a n$. We lose no generality by assuming that for all $a, b \in \Sigma$, if $a \neq b$, then $(V_a - \Sigma_a) \cap (V_b - \Sigma_b) = (V_a - \Sigma_a) \cap (V - \Sigma) = \phi$.

Let $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ be a set of new symbols. Let $h : V^* \to ((V - \Sigma) \cup \bar{\Sigma})^*$ be the homomorphism determined by defining $h(Z) = Z$ for $Z \in V - \Sigma$ and $h(a) = \bar{a}$ for $a \in \Sigma$. Let $R_1 = \{h(\rho) \to h(\theta) \mid \rho \to \theta \in R\} \cup \{\bar{a} \to X_a \mid a \in \Sigma\} \cup (\bigcup_{a\in\Sigma} R_a)$, let $V_1 = V \cup \bar{\Sigma} \cup (\bigcup_{a\in\Sigma} V_a)$, and let $\Sigma_1 = \bigcup_{a\in\Sigma} \Sigma_a$. Let $G_1 = (V_1, \Sigma_1, R_1, X_1)$.

We would like to say that $L(G_1) = \tau[L(G)]$. However, rules must be added to ensure that the following situation does not occur. If for some $a \in \Sigma$, $y, z \in V^*$, $X \overset{*}{\Rightarrow} yaaz$ in G, then $X \overset{*}{\Rightarrow} yX_aX_az$ in G_1. It is possible that $\{w \in \Sigma_a^* \mid X_aX_a \overset{*}{\Rightarrow} w \text{ in } G_a\}$ contains more than $L(G_a)L(G_a) = \tau(a)\tau(a)$. Hence we must add rules to R_1 to see that this does not occur. This may be accomplished by making "copies" of each G_a and adding rules to ensure that strings of the form yX_aX_az do not occur. The details may be found in [1].

Let $c_1 = \max\{c, c_a \mid a \in \Sigma\}$. Then it is clear that $cf(x) + (c_1 + 1)x$ bounds G_1. Further, the modification of G_1 described above yields a grammar bounded by $(c + c_1 + 1)f$. Thus $\tau[L(G)] \in \mathscr{L}_{\text{CS}}(f)$ if G is CS and $\tau[L(G)] \in \mathscr{L}(f)$ if G is arbitrary.

Corollary 5.6. *If f is a bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under e-free regular substitution.*

Corollary 5.7. *If f is a bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under nonerasing homomorphism.*[9]

The families $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under e-free substitution if f meets a simple condition.

Definition 5.8. A function f is superadditive if for all $x, y \geqslant 0$, $f(x) + f(y) \leqslant f(x + y)$.

Theorem 5.9. *If f is a superadditive bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under e-free substitution.*

Corollary 5.10. *If f is a superadditive bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under Kleene $+$.*

Proof. Since $\{a\}$ is an e-free regular set, $\{a\}^+ \in \text{LINEAR}_{\text{CS}} \subseteq \mathscr{L}_{\text{CS}}(f) \subseteq \mathscr{L}(f)$. For any $L \in \mathscr{L}_{\text{CS}}(f)$, let τ be the $\mathscr{L}_{\text{CS}}(f)$-substitution on $\{a\}$ defined by $\tau(a) = L$. Then $\tau(\{a\}^+) = \bigcup_{w\in\{a\}^+} \tau(w) = L^+$. Since $\mathscr{L}_{\text{CS}}(f)$ is closed under e-free substitution, $L^+ = \tau(\{a\}^+) \subseteq \mathscr{L}_{\text{CS}}(f)$. The same argument applies to $\mathscr{L}(f)$.

We now investigate closure under inverse homomorphism.[10] To do this we must

[9] A homomorphism $h : \Sigma^* \to \Delta^*$ is *nonerasing* if $h(w) = e$ implies $w = e$.

[10] A family $\mathscr{F}$ of languages is closed under inverse homomorphism if $L \in \mathscr{F}$ and h a homomorphism imply $h^{-1}[L] = \{w \mid h(w) \in L\} \in \mathscr{F}$.

introduce new functions and new families of languages. These results closely parallel those of [3] for families of languages accepted by time-bounded Turing acceptors and for images of AFLs under bounded erasing.

Notation. For any function f and any integer $k > 0$, define the function f_k by $f_k(x) = f(kx)$. For any bounding function f, let $\mathscr{S}(\mathscr{L}_{\text{CS}}(f)) = \bigcup_k \mathscr{L}_{\text{CS}}(f_k)$ and $\mathscr{S}(\mathscr{L}(f)) = \bigcup_k \mathscr{L}(f_k)$.

DEFINITION 5.11. A function f is *semihomogeneous* if for every $k_1 > 0$, there is a $k_2 > 0$ such that for all $x \geqslant 0, f(k_1x) \leqslant k_2f(x)$.

It is straightforward to verify that if f is semihomogeneous (superadditive), then for any integer $k > 0$, f_k is semihomogeneous (superadditive). Also note that a nondecreasing function f is semihomogeneous if and only if there is a $k_1 \geqslant 1$ such that for some $k_2 > 0$ and all $x > 0, f(k_1x) \leqslant k_2f(x)$.

LEMMA 5.12. *If f is a semihomogeneous bounding function, then $\mathscr{S}(\mathscr{L}_{\text{CS}}(f)) = \mathscr{L}_{\text{CS}}(f)$ (and $\mathscr{S}(\mathscr{L}(f)) = \mathscr{L}(f)$).*

Proof. For any $k_1 > 0$, there is a $k_2 > 0$ such that for all $x \geqslant 0, f_{k_1}(x) = f(k_1x) \leqslant k_2f(x)$. Thus, $\mathscr{L}_{\text{CS}}(f_{k_1}) \subseteq \mathscr{L}_{\text{CS}}(k_2f) = \mathscr{L}_{\text{CS}}(f)$ so that $\mathscr{S}(\mathscr{L}_{\text{CS}}(f)) = \bigcup_k \mathscr{L}_{\text{CS}}(f_k) \subseteq \mathscr{L}_{\text{CS}}(f)$. But $\mathscr{L}_{\text{CS}}(f) = \mathscr{L}_{\text{CS}}(f_1) \subseteq \bigcup_k \mathscr{L}_{\text{CS}}(f_k)$. Hence, $\mathscr{S}(\mathscr{L}_{\text{CS}}(f)) = \mathscr{L}_{\text{CS}}(f)$.

Note that many functions arising in the study of computational complexity of formal languages are both superadditive and semihomogeneous, e.g., $f(x) = x \log x$, $f(x) = x^k$ for $k \geqslant 1$. Exponential functions are not semihomogeneous. In [11], it is shown that families of languages accepted by Turing acceptors which operate within tape bound $f(x) = 2^x$ are not closed under inverse homomorphism. We cannot show such strong results here since we do not know, for example, whether $\mathscr{L}(2^x)$ and $\mathscr{L}(4^x)$ are different families. Here we parallel results in [3] for the families of languages defined by time-bounded Turing acceptors by showing that $\mathscr{L}_{\text{CS}}(f)$ is closed under inverse homomorphism if and only if $\mathscr{L}_{\text{CS}}(f) = \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ (5.21), and that the smallest AFL containing $\mathscr{L}_{\text{CS}}(f)$ is the smallest AFL containing $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ (5.25).

In order to show that $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and $\mathscr{S}(\mathscr{L}(f))$ are closed under inverse homomorphism, we shall show that these families are closed under limited erasing.[11] We follow the arguments of [5].

LEMMA 5.13. *If f is a bounding function, then both $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and $\mathscr{S}(\mathscr{L}(f))$ are closed under union, concatenation, reversal, intersection with regular sets,* LINEAR$_{\text{CS}}$-*substitution, regular substitution, and nonerasing homomorphism. If f is superadditive, $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and $\mathscr{S}(\mathscr{L}(f))$ are also closed under substitution and Kleene $+$.*

[11] A homomorphism $h : \Sigma^* \to \Delta^*$ is *k-limited on L*, $k > 0$ an integer and $L \subseteq \Sigma^*$, if $e \notin h[L]$ and if for all $w \in L$ and all $x, y, z \in \Sigma^*$ such that $w = xyz$, $h(y) = e$ implies $|y| \leqslant k$. A family $\mathscr{F}$ of languages is closed under *limited erasing* if $L \in \mathscr{F}$ and h k-limited on L imply $h[L] \in \mathscr{F}$.

Proof. If f is a bounding function and k a positive integer, f_k is also a bounding function, and so $\mathscr{L}_{\text{CS}}(f_k)$ has each of the desired closure properties. Since $\mathscr{S}(\mathscr{L}_{\text{CS}}(f)) = \bigcup_k \mathscr{L}_{\text{CS}}(f_k)$ and $\mathscr{L}_{\text{CS}}(f_k) \subsetneq \mathscr{L}_{\text{CS}}(f_{k+1})$, $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ is the union of an increasing chain of families, each of which has the desired properties. Thus $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ has the desired closure properties. Similarly, $\mathscr{S}(\mathscr{L}(f))$ has the desired closure properties.

Notation. Let Σ be an alphabet and c a symbol not in Σ. Let $\Sigma_c = \Sigma \cup \{c\}$. Let $\mu_c : \Sigma_c{}^* \to \Sigma^*$ be the homomorphism determined by defining $\mu_c(a) = a$ for $a \in \Sigma$ and $\mu_c(c) = e$.

LEMMA 5.14. *Let f be a bounding function and $L \in \mathscr{L}_{\text{CS}}(f)$. If $L \subsetneq \Sigma^*(c\Sigma\Sigma^*)^*$ where c is not in Σ, then $\mu_c[L] \in \mathscr{L}_{\text{CS}}(f_3)$.*

Proof. Notice that $L \subsetneq \Sigma^*(c\Sigma\Sigma^*)^*$ where $c \notin \Sigma$ implies that $c \notin L$ and that no word in L contains two consecutive c's.

Let $G = (V, \Sigma_c, R, X)$ be a CS grammar such that $L(G) = L$ and f bounds G. Assume that each rule in R is of one of the forms

$$\left.\begin{aligned} Z_1Z_2 &\to Y_1Y_2 \\ Z_1 &\to Y_1Y_2 \\ Z_1 &\to a \end{aligned}\right\} Z_1, Z_2, Y_1, Y_2 \in V - \Sigma_c,\ a \in \Sigma_c .$$

Let k be the least positive integer such that $T_G(n)\downarrow$ implies $T_G(n) \leqslant kf(n)$.

Let

$$\begin{aligned} V_1 &= V \cup (V \times V) \cup (V \times V \times V), \\ \Sigma_1 &= \Sigma_c \cup (\Sigma_c \times \Sigma_c) \cup (\Sigma_c \times \Sigma_c \times \Sigma_c), \end{aligned}$$

and let $h : V_1{}^* \to V^*$ be the homomorphism determined by defining

$$\left.\begin{aligned} h(v_1) &= v_1 \\ h((v_1, v_2)) &= v_1v_2 \\ h((v_1, v_2, v_3)) &= v_1v_2v_3 \end{aligned}\right\} v_1, v_2, v_3 \in V.$$

Trivially, h is nonerasing and surjective, and $h^{-1}[\Sigma_c{}^*] = \Sigma_1{}^*$. Also, for any $w \in V_1{}^*$, $(1/3)\,|\,h(w)| \leqslant |\,w\,| \leqslant |\,h(w)|$.

Let $g : \Sigma_1{}^* \to \Sigma_c{}^*$ be the homomorphism determined by defining

$$\begin{aligned} g(a) &= a \\ g((a, b)) &= ab \\ g((a, b, d)) &= abd \\ g((a, c)) &= g((c, a)) = g((a, c, c)) = g((c, a, c)) = g((c, c, a)) = a \\ g((a, b, c)) &= g((a, c, b)) = g((c, a, b)) = ab \\ g(c) &= g((c, c)) = g((c, c, c)) = c \end{aligned}$$

for all $a, b, d \in \Sigma$. Then g is nonerasing, $g(w) \in \Sigma^*$ if and only if

$$w \in (\Sigma_1 - \{c, (c, c), (c, c, c)\})^*,$$

and $g(w) \in \{c\}^*$ if and only if $w \in \{c, (c, c), (c, c, c)\}^*$. Thus, for any

$$L' \subseteq (\Sigma_1 - \{c, (c, c), (c, c, c)\})^*, \qquad g[L'] = \mu_c[h[L']].$$

We now construct a CS grammar G_0 such that $h[L(G_0)] = L(G) = L$, $g[L(G_0)] = \mu_c[h[L(G_0)]] = \mu_c[L]$, and f_3 bounds G_0.

Construct the following sets of rules:

$$\begin{aligned} R_1 = \{& Z_1 \to (Y_1, Y_2), (Z_1, Z_2) \to (Y_1, Y_2, Z_2), (Z_2, Z_1) \to (Z_2, Y_1, Y_2),\\ & (Z_2, Z_1, Z_3) \to (Z_2, Y_1)(Y_2, Z_3), (Z_2, Z_3, Z_1) \to (Z_2, Z_3)(Y_1, Y_2),\\ & (Z_1, Z_2, Z_3) \to (Y_1, Y_2)(Z_2, Z_3) \mid Z_1 \to Y_1Y_2 \in R, Z_2, Z_3 \in V - \Sigma_c\}; \end{aligned}$$

$$\begin{aligned} R_2 = \{& (Z_1, Z_2) \to (Y_1, Y_2), (Z_1, Z_2, Z_3) \to (Y_1, Y_2, Z_3),\\ & (Z_3, Z_1, Z_2) \to (Z_3, Y_1, Y_2), (Z_3, Z_1)(Z_2, Z_4) \to (Z_3, Y_1)(Y_2, Z_4),\\ & (Z_3, Z_4, Z_1)(Z_2, Z_5) \to (Z_3, Z_4, Y_1)(Y_2, Z_5),\\ & (Z_3, Z_1)(Z_2, Z_4, Z_5) \to (Z_3, Y_1)(Y_2, Z_4, Z_5),\\ & (Z_3, Z_4, Z_1)(Z_2, Z_5, Z_6) \to (Z_3, Z_4, Y_1)(Y_2, Z_5, Z_6) \mid\\ & Z_1Z_2 \to Y_1Y_2 \in R, Z_3, Z_4, Z_5, Z_6 \in V - \Sigma_c\}; \end{aligned}$$

$$\begin{aligned} R_3 = \{& Z_1 \to a, (Z_1, Z_2) \to (a, Z_2), (Z_2, Z_1) \to (Z_2, a),\\ & (Z_3, Z_2, Z_1) \to (Z_3, Z_2, a), (Z_3, Z_1, Z_2) \to (Z_3, a, Z_2),\\ & (Z_1, Z_2, Z_3) \to (a, Z_2, Z_3) \mid Z_1 \to a \in R, Z_2, Z_3 \in V\}. \end{aligned}$$

Let $R_0 = R_1 \cup R_2 \cup R_3$ and $G_0 = (V_1, \Sigma_1, R_0, X)$. Then G_0 is a CS grammar and clearly $h[L(G_0, V_1^+)] = L(G, V^+)$ since G_0 simply imitates the derivations of G within the "coded" vocabulary V_1 and h "decodes" V_1^* to V^*. By construction of R_0, it is clear that $L(G_0) \subseteq \Sigma_c \cup ((\Sigma_c \times \Sigma_c) \cup (\Sigma_c \times \Sigma_c \times \Sigma_c))^*$. Further, if $w \in V^*$ and $X \overset{*}{\Rightarrow} w$ in G, then there exists $w_1 \in h^{-1}(w)$ such that $X \overset{*}{\Rightarrow} w_1$ in G_0 and $t_{G_0}(w_1) = t_G(w)$; conversely, if $w \in V_1^*$ and $X \overset{*}{\Rightarrow} w$ in G_0, then $X \overset{*}{\Rightarrow} h(w)$ in G and $t_G(h(w)) = t_{G_0}(w)$. Since for any $w \in V_1^*$, $(1/3)|h(w)| \leqslant |w| \leqslant |h(w)|$ and for $w \in L(G_0, V^+)$, $t_G(h(w)) = t_{G_0}(w)$, we have $T_{G_0}(n) \leqslant T_G(3n)$. Since $T_G(n)\downarrow$ implies $T_G(n) \leqslant kf(n)$, $T_G(3n) \leqslant kf(3n) = kf_3(n)$. Thus, kf_3 bounds G_0; so $L(G_0) \in \mathscr{L}(f_3)$.

Since $h[L(G_0, V_1^+)] = L(G, V^+)$, $h[L(G_0)] = L(G) = L$. Since

$$L(G_0) \subseteq \Sigma_c \cup ((\Sigma_c \times \Sigma_c) \cup (\Sigma_c \times \Sigma_c \times \Sigma_c))^*$$

and $h[L(G_0)] = L$, the hypothesis that $L \subseteq \Sigma^*(c\Sigma\Sigma^*)^*$ implies that no word of $L(G_0)$ contains any occurrence of (c, c) or (c, c, c) and that $c \notin L(G_0)$. Thus,

$$L(G_0) \subseteq (\Sigma_1 - \{c, (c, c), (c, c, c)\})^*$$

so that $g[L(G_0)] = \mu_c[h[L(G_0)]] = \mu_c[L]$. Since g is nonerasing and $L(G_0) \in \mathscr{L}_{\text{CS}}(f_3)$, $\mu_c[L] = g[L(G_0)] \in \mathscr{L}_{\text{CS}}(f_3)$.

THEOREM 5.15. *If f is a bounding function, then $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ is closed under limited erasing.*

Proof. Let $L \in \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and let $h : \Sigma^* \to \Sigma^*$ be a homomorphism that is k-limited on L. If Σ is an alphabet such that $L \subseteq \Sigma^*$ and $c \notin \Sigma$, let $g : \Sigma^* \to \Sigma_c{}^*$ be the homomorphism determined by defining $g(a) = h(a)$ if $h(a) \neq e$ and $g(a) = c$ if $h(a) = e$. Then g is nonerasing; so $g[L] \in \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and also $\mu_c[g[L]] = h[L]$ so that μ_c is k-limited on $g[L]$. Since $e \notin h[L]$, $g[L] \cap \{c\}^+ = \phi$. Thus it is enough to show that $\mu_c[g[L]] \in \mathscr{S}(\mathscr{L}(f))$.

Let $L_1 = g[L]$. Since $L_1 \in \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ there is an integer q such that $L_1 \in \mathscr{L}_{\text{CS}}(f_q)$, and since $\mathscr{S}(\mathscr{L}_{\text{CS}}(f_q)) = \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$, we lose no generality by assuming that $q = 1$.

Let d be a symbol not in Σ_c. Let τ be the regular substitution on Σ_c defined by $\tau(a) = \{a\}$ for $a \in \Sigma$ and $\tau(c) = \{c, d\}$. Let $R = \Sigma^* \cup \Sigma^*(d\Sigma\Sigma^*)^* \cup \Sigma^*(dc\Sigma^*)^*$ and $L_2 = \tau[L_1] \cap R$. Since R is regular and τ is a regular substitution, $L_2 \in \mathscr{L}_{\text{CS}}(f)$. But $L_2 \subseteq \Sigma_c{}^*(d\Sigma_c\Sigma_c{}^*)^*$. Let $\nu_d : (\Sigma_c \cup \{d\})^* \to \Sigma_c{}^*$ be the homomorphism determined by defining $\nu_d(a) = a$ for $a \in \Sigma_c$ and $\nu_c(d) = e$. By 5.14, $\nu_d[L_2] \in \mathscr{S}(\mathscr{L}(f))$. Clearly $\mu_c[\nu_d[L_2]] = \mu_c[L_1] = h[L]$, and since no word in $\nu_d[L_2]$ or in L_2 has a substring of k or more c's, μ_c is $k - 1$ limited on $\nu_d[L_2]$. Thus, applying the regular substitution τ, the intersection with regular set R, and the homomorphism ν_d in sequence at most k times will yield $\mu_c[L_1] = h[L]$. Thus $h[L] \in \mathscr{S}(\mathscr{L}(f))$.

THEOREM 5.16. *If f is a bounding function, then $\mathscr{S}(\mathscr{L}(f))$ is closed under linear erasing.*[12]

Proof. Since $\mathscr{S}(\mathscr{L}(f)) = \bigcup_k \mathscr{L}(f_k)$ and for any k, $\mathscr{S}(\mathscr{L}(f_k)) = \mathscr{S}(\mathscr{L}(f))$, it is sufficient to show that the image of $\mathscr{L}(f)$ under linear erasing is included in $\mathscr{S}(\mathscr{L}(f))$. Let $G = (V, \Sigma, R, X)$ be a grammar such that f bounds G and such that $\rho \to \theta \in R$ implies $\rho \in (V - \Sigma)^*$. Let $h : \Sigma^* \to \Sigma^*$ be a homomorphism and k a positive integer such that for all $w \in L(G)$, $|w| \leqslant k\,|h(w)|$.

Let $g : V^* \to V^*$ be the homomorphism determined by defining $g(Z) = Z$ for $Z \in V - \Sigma$ and $g(a) = h(a)$ for $a \in \Sigma$. Since g extends h to V^*, $|w| \leqslant k\,|g(w)|$ for all $w \in V^*$. Let $R_0 = \{g(\rho) \to g(\theta) \mid \rho \to \theta \in R\}$ and let $G_0 = (V, \Sigma, R_0, X)$. Then $L(G_0) = h[L(G)]$. For any

$$w \in L(G, V^+), \qquad t_{G_0}(g(w)) \leqslant t_G(w) \leqslant T_G(|w|) \leqslant T_G(k\,|g(w)|) \leqslant f(k\,|g(w)|),$$

and for any $w' \in L(G_0, V^+)$ there exists $w \in g^{-1}(w')$ such that $t_{G_0}(w') \leqslant t_G(w) \leqslant T_G(|w|) \leqslant T_G(k\,|g(w)|) \leqslant f(k\,|g(w)|)$. Thus $T_{G_0}(n) \leqslant T_G(kn) \leqslant f(kn) = f_k(n)$, so f_k bounds G_0 and $h[L(G)] = L(G_0) \in \mathscr{L}(f_k) \subseteq \mathscr{S}(\mathscr{L}(f))$.

[12] A homomorphism $h : \Sigma^* \to \Delta^*$ is *linear-erasing on L* for $L \subseteq \Sigma^*$ if there exists an integer $k \geqslant 1$ such that for all $w \in L$, $|w| \leqslant k\,|h(w)|$. A family $\mathscr{F}$ of languages is closed under *linear erasing* if $L \in \mathscr{F}$ and h linear-erasing on L imply $h[L] \in \mathscr{F}$.

COROLLARY 5.17. *If f is a bounding function, then $\mathscr{S}(\mathscr{L}(f))$ is closed under limited erasing.*

COROLLARY 5.18. *If f is a bounding function, then the closure of $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ under linear erasing is included in $\mathscr{S}(\mathscr{L}(f))$.*

THEOREM 5.19. *If f is a bounding function, then $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and $\mathscr{S}(\mathscr{L}(f))$ are closed under inverse homomorphism.*

Proof. In [11], it is shown that for any family $\mathscr{F}$ of e-free languages, if $\mathscr{F}$ is closed under regular substitution, limited erasing, and union and intersection with e-free regular sets, then $\mathscr{F}$ is closed under inverse homomorphism. Hence the above lemmas yield the result.

COROLLARY 5.20. *If f is a semihomogeneous bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under inverse homomorphism.*

We shall now show that if $\mathscr{L}_{\text{CS}}(f)$ is closed under inverse homomorphism, then $\mathscr{L}_{\text{CS}}(f) = \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$.

THEOREM 5.21. *If f is a bounding function, then $\mathscr{L}_{\text{CS}}(f)$* (resp., *$\mathscr{L}(f)$*) *is closed under inverse homomorphism if and only if $\mathscr{L}_{\text{CS}}(f) = \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$* (resp., *$\mathscr{L}(f) = \mathscr{S}(\mathscr{L}(f))$*).

Proof. Since $\mathscr{S}(\mathscr{L}_{\text{CS}}(f)) = \bigcup_k \mathscr{L}_{\text{CS}}(f_k)$, and for any k, $\mathscr{S}(\mathscr{L}_{\text{CS}}(f_k)) = \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$, it is sufficient to show that if $\mathscr{L}_{\text{CS}}(f)$ is closed under inverse homomorphism, then $\mathscr{L}_{\text{CS}}(f) = \mathscr{L}_{\text{CS}}(f_2)$. Thus consider $L \in \mathscr{L}_{\text{CS}}(f_2)$ where $L \subsetneq \Sigma^*$. We shall show that $L \in \mathscr{L}_{\text{CS}}(f)$.

Let $G = (V, \Sigma, R, X)$ be a CS grammar such that f_2 bounds G and $L(G) = L$. Let q be the least integer such that $T_G(n)\downarrow$ implies $T_G(n) \leqslant q f_2(n)$. Then for any $w \in L(G, V^+)$, there is a proper derivation of w in G of length no greater than $q f_2(|w|) = q f(2|w|)$.

Let X' and c be symbols not in V, let $V_c = V \cup \{c, X'\}$, and let

$$L_c = \{a_1 c \cdots a_n c \mid a_i \in \Sigma, a_1 \cdots a_n \in L\}.$$

Let $h : V^* \to V_c^*$ be the homomorphism determined by definining $h(Z) = Zc$ for each $Z \in V$. Let $R' = \{X \to h(\theta) \mid X \to \theta \in R\} \cup \{h(\rho) \to h(\theta) \mid \rho \to \theta \in R\}$. Let $G' = (V_c, \Sigma_c, R', X')$. Then, it is immediate that $L(G') = L_c$, G' is CS, and $h[L(G, V^+)] = L(G', V_c^+)$. Further, if $w \in L(G', V_c^+)$ then $|w|$ is even and there is a proper derivation of w in G of length no greater than $q f_2(|w|/2) = q f(2|w|/2) = q f(|w|)$ so that $q f$ bounds G'. Thus, $L_c = L(G') \in \mathscr{L}_{\text{CS}}(f)$. But $L = h^{-1}[L_c]$ and $\mathscr{L}_{\text{CS}}(f)$ is closed under inverse homomorphism so that $L \in \mathscr{L}_{\text{CS}}(f)$.

We summarize the positive closure results so far obtained by formally defining AFLs [6] and stating these results in terms of AFLs.

DEFINITION 5.22. An *abstract family of languages* (AFL) is a family containing at least one nonempty set and closed under union, concatenation, Kleene $+$, intersection with regular sets, nonerasing homomorphism and inverse homomorphism. An AFL is *full* if it is closed under arbitrary homomorphism, and is *e-free* if every language in it is e-free.

THEOREM 5.23. *If f is a superadditive bounding function, then $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ and $\mathscr{S}(\mathscr{L}(f))$ are e-free* AFLs, *closed under reversal and substitution, but are not full* AFLs. *Also, $\mathscr{S}(\mathscr{L}(f))$ is closed under linear erasing.*

COROLLARY 5.24. *If f is a superadditive semihomogeneous bounding function, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are e-free* AFLs, *closed under reversal and substitution, but are not full* AFLs. *Also, $\mathscr{L}(f)$ is closed under linear erasing.*

From the proof of Theorem 5.21, we see that $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ is included in the closure of $\mathscr{L}_{\text{CS}}(f)$ under inverse homomorphism. Hence we have the following corollary.

COROLLARY 5.25. *For any bounding function f, the smallest* AFL *containing $\mathscr{L}_{\text{CS}}(f)$* (resp., $\mathscr{L}(f)$) *is the smallest* AFL *containing $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$* (resp., $\mathscr{S}(\mathscr{L}(f))$).

We now establish one more closure property, that $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under intersection if f is "at least quadratic." This is not an AFL property, since there are AFL's and full AFL's which are not closed under intersection. In Section 6, we show that we cannot relax the condition of being "at least quadratic."

THEOREM 5.26. *If f is a bounding function such that for some $c > 0$ and all x, $f(x) \geqslant cx^2$, then $\mathscr{L}_{\text{CS}}(f)$ and $\mathscr{L}(f)$ are closed under intersection.*

Sketch of the Proof. For $i = 1, 2$, let $G_i = (V_i, \Sigma_i, R_i, X_i)$ be CS grammars bounded by f. Let X and Z be new symbols not in $V_1 \cup V_2$. Let

$$V_3 = \{X\} \cup (V_1 \times V_2) \cup (V_1 \times \{Z\}) \cup \Sigma_1 \cup \Sigma_2 .$$

One can construct a CS grammar $G_3 = (V_3, \Sigma_1 \cap \Sigma_2, R_3, X)$ which generates strings in $\Sigma_1 \cap \Sigma_2$ by first imitating a derivation of G_1 on the "first-coordinates" of symbols in $(V_1 \times \{Z\})$ and then imitating a derivation of G_2 on the "second coordinates" of the symbols of strings in $(\Sigma_1 \times \{X_2\})(\Sigma_1 \times (\{Z\} \cup V_2))^*$. If both derivations give the same terminal string, that string can be generated by rules of the form $(a, a) \to a$. Since f is "at least quadratic", the number of steps necessary to imitate length increasing steps in a derivation in G_2 will not majorize f. Thus the construction is essentially that used in proving by means of CS grammars that the family of CS languages is closed under intersection.

Corollary 5.27. *If f is a bounding function such that for some $c > 0$ and all x, $f(x) \geqslant cx^2$, then $\mathcal{S}(\mathcal{L}_{\mathrm{CS}}(f))$ and $\mathcal{S}(\mathcal{L}(f))$ are closed under intersection.*

The question of whether $\mathrm{LINEAR}_{\mathrm{CS}}$ is closed under linear erasing remains open. However, the methods used to prove Lemma 5.16, Theorem 5.17, and Theorem 5.26 may be used to show the following propositions.

Proposition 5.28. *Let Σ be a finite alphabet and $c \notin \Sigma$. Suppose $L \subseteq \Sigma^* c^*$, $L \in \mathrm{LINEAR}_{\mathrm{CS}}$, and $k > 0$ is an integer such that each $w \in L$ is of the form $w = w_1 c^t$ where $w_1 \in \Sigma^*$ and $t \leqslant k \sqrt{|w_1|}$. Then $\mu_c[L] \in \mathrm{LINEAR}_{\mathrm{CS}}$.*

Proposition 5.29. *If f is a bounding function with the property that for some $k > 0$ and all x, $kf(x) \geqslant x^2$, then $\mathcal{S}(\mathcal{L}_{\mathrm{CS}}(f))$ is closed under linear erasing.*

6. Negative Closure Properties and Undecidability Results

In this section we establish some negative closure results of the families $\mathcal{L}_{\mathrm{CS}}(f)$ and $\mathcal{L}(f)$, particularly for the families $\mathrm{LINEAR}_{\mathrm{CS}}$ and LINEAR. We also establish various undecidability results for the family $\mathrm{LINEAR}_{\mathrm{CS}}$ and thus for all families $\mathcal{L}_{\mathrm{CS}}(f)$ and $\mathcal{L}(f)$.

The most interesting negative closure result is due to Gladkii [7].

Theorem 6.1. *Let $\psi : \{a, b\}^* \to \{a, b\}^*$ be any function with the property that there is a constant $k > 0$ such that for all $w \in \{a, b\}^*$, $|\psi(w)| \leqslant k \mid w \mid$. Let*

$$L_\psi = \{wc\psi(w)\, cw \mid w \in \{a, b\}^*\}.$$

If f is any bounding function such that $\lim_{n\to\infty} f(n)/n^2 = 0$, then $L_\psi \notin \mathcal{L}_{\mathrm{CS}}(f)$.

While in [7] this theorem is proved only for the case of CS grammars, it can be extended to arbitrary grammars [1]. This theorem is most useful in establishing the following.

Theorem 6.2. *If f is any bounding function such that $\lim_{n\to\infty} f(n)/n^2 = 0$, then $\mathcal{L}_{\mathrm{CS}}(f)$ is not closed under intersection. In particular, $\mathrm{LINEAR}_{\mathrm{CS}}$ is not closed under intersection.*

Proof. Let $L_1 = \{wcw^R cw' \mid w, w' \in \{a, b\}^*\}$ and $L_2 = \{w'cwcw^R \mid w, w' \in \{a, b\}^*\}$. Clearly, L_1 and L_2 are deterministic context-free languages so that

$$L_1, L_2 \in \mathrm{LINEAR}_{\mathrm{CS}} \subseteq \mathcal{L}_{\mathrm{CS}}(f).$$

Let $\psi : \{a, b\}^* \to \{a, b\}^*$ be defined by $\psi(w) = w^R$ for all $w \in \{a, b\}^*$, so that

$|\psi(w)| = |w|$ for all $w \in \{a, b\}^*$. Then $L_\psi = \{wcw^Rcw \mid w \in \{a, b\}^*\} = L_1 \cap L_2$. By Gladkii's result, $L_\psi \notin \mathscr{L}_{CS}(f)$ so that $L_1 \cap L_2 \notin \mathscr{L}_{CS}(f)$.

Corollary 6.3. *If f is any bounding function such that* $\lim_{n\to\infty} f(n)/n^2 = 0$, *then $\mathscr{L}_{CS}(f)$ is not closed under complementation.*

Corollary 6.4. *If f is any bounding function such that* $\lim_{n\to\infty} f(n)/n^2 = 0$, *then $\mathscr{L}_{CS}(f)$ is not closed under intersection with deterministic context-free languages.*

The above results also apply to the appropriate families $\mathscr{L}(f)$ and $\mathscr{S}(\mathscr{L}(f))$.

In proving Lemma 5.2, we showed that for each Type 0 grammar G_1, one can construct a monotonic grammar G_2 such that each rule of G_2 is either a length-increasing rule or a length-preserving rule which generates a terminal symbol, and such that if $L(G_1) \subseteq \Sigma^*$, then $L(G_2) \subseteq \{d\}^* L(G_1)$ and $\mu_d[L(G_2)] = L(G_1)$, where $d \notin \Sigma$ and $\mu_d : (\Sigma \cup \{d\})^* \to \Sigma^*$ is the homomorphism determined by defining $\mu_d(a) = a$ for $a \in \Sigma$ and $\mu_d(d) = e$. The restrictions on the form of the rules of G_2 are sufficient to force G_2 to be bounded by the function $f(x) = 2x$, so that G_2 is a linear-time grammar. From the results in Section 5, from G_2 one can construct a linear-time monotonic grammar G_3 such that $L(G_3) = \{d\}^* L(G_2)$ (and so $\mu_d[L(G_3)] = L(G_1)$). We use this fact to establish the following results.

Theorem 6.5. *If f is any bounding function, then neither $\mathscr{L}_{CS}(f)$ nor $\mathscr{L}(f)$ is closed under arbitrary homomorphic mappings.*

Proof. The image of LINEAR_{CS} under arbitrary homomorphic mappings is the family of recursively enumerable sets. But $\mathscr{L}_{CS}(f)$ and $\mathscr{L}(f)$ are families of recursive sets and LINEAR_{CS} is a subfamily of both of these families.

Theorem 6.6. *The question "is $L = \phi$?" is undecidable for $L \in \text{LINEAR}_{CS}$.*

Proof. As above, for every Type 0 grammar G_1 (hence, for every recursively enumerable set $L(G_1)$), one can construct a linear-time grammar G_2 such that $\mu_d[L(G_2)] = L(G_1)$. Now $L(G_2) = \phi$ if and only if $L(G_1) = \phi$, since $\mu_d[L(G_2)] = L(G_1)$. But for Type 0 grammars G, it is undecidable whether $L(G) = \phi$; hence, this question is undecidable for linear-time grammars.

Corollary 6.7. *Each of the following questions is undecidable for $L \in \text{LINEAR}_{CS}$ and R a regular set, where $L \subseteq \Sigma^+$:*

(i) *is $L \subseteq R$?*
(ii) *is $\Sigma^+ - L = \Sigma^+$?*
(iii) *is $R \subseteq \Sigma^+ - L$?*
(iv) *is $\Sigma^+ - L \subseteq R$?*
(v) *is $\Sigma^+ - L = R$?*

THEOREM 6.8. *The question "is L finite?" is undecidable for $L \in \mathrm{LINEAR}_{\mathrm{CS}}$.*

Proof. As above, for every Type 0 grammar G_1, one can construct a linear-time grammar G_3 such that $\mu_d[L(G_3)] = L(G_1)$ and such that $L(G_3)$ is finite if and only if $L(G_3) = \phi$. But $L(G_3) = \phi$ if and only if $L(G_1) = \phi$. Hence, it is undecidable whether $L(G_3)$ is finite.

For any bounding function f, $\mathrm{LINEAR}_{\mathrm{CS}} \subseteq \mathscr{L}_{\mathrm{CS}}(f) \subseteq \mathscr{L}(f)$ so that the undecidability properties of $\mathrm{LINEAR}_{\mathrm{CS}}$ apply to the families $\mathscr{L}_{\mathrm{CS}}(f)$ and $\mathscr{L}(f)$. Recall that every e-free CF language is in $\mathrm{LINEAR}_{\mathrm{CS}}$ so that any question that is undecidable for the CF languages is undecidable for languages in $\mathrm{LINEAR}_{\mathrm{CS}}$. (For a discussion of undecidability properties of the CF languages, see [4], [10], or [15].)

Consider the construction of linear-time grammars from Type 0 grammars as indicated above. Let $\mathscr{L}$ be the smallest AFL containing the languages generated by those linear-time grammars. Then $\mathscr{L}$ is an "effective family of languages" and an "effective AFL" in the sense of Greibach [10], since the proofs of the positive closure results in Section 5 all involved effective constructions. Further, the proofs of Theorems 6.6 and 6.8 and Corollary 6.7 show that those results also apply to $\mathscr{L}$. Since $\mathscr{L}$ is a subfamily of $\mathrm{LINEAR}_{\mathrm{CS}}$, the results of [10] can be applied to prove the following theorem.

THEOREM 6.9. *The question "is L CF?" is undecidable for $L \in \mathrm{LINEAR}_{\mathrm{CS}}$.*

7. HIERARCHY RESULTS

The languages studied in this paper are defined by time-bounded grammars. In Section 4, it was shown that if a language is generated by a grammar bounded by a function f, then the language is accepted by a nondeterministic Turing acceptor which accepts within time bound f. In this section we study the relationships between the families $\mathscr{L}_{\mathrm{CS}}(f)$ (and $\mathscr{L}(f)$) and certain families of formal languages which have been studied extensively in the literature, particularly those defined by time-bounded acceptors.

DEFINITION 7.1. A language L is *quasi-realtime* if there is a nondeterministic multitape Turing acceptor M such that $L(M) = L$ and such that M accepts within time bound i.[13] A language L is *real-time definable* if there is a deterministic multitape Turing acceptor M such that $L(M) = L$ and such that M accepts within time bound i.

THEOREM 7.2. *The families* $\mathrm{LINEAR}_{\mathrm{CS}}$ *and* LINEAR *are proper subfamilies of the quasi realtime languages and are not comparable to the real-time definable languages.*

[13] Again, i is the identity function.

Proof. By Theorem 4.2, if $L \in$ LINEAR, then L is accepted by a nondeterministic multitape Turing acceptor which accepts in linear time. In [1] and [2], it is shown that linear time is no more powerful than quasi realtime, so that every language in LINEAR is a quasi realtime language. In [2], it is shown that the quasi realtime languages are closed under intersection, but the results of Section 6 show that neither $\text{LINEAR}_{\text{CS}}$ nor LINEAR are closed under intersection. Hence the containment is proper.

In [19] it is shown that there are CF languages which are not real-time definable. Since the CF languages are in $\text{LINEAR}_{\text{CS}}$, there are languages in $\text{LINEAR}_{\text{CS}}$ which are not real-time definable. The language $\{wcw^Rcw \mid w \in \{a, b\}^*\}$ is clearly real-time definable, but, as shown in the proof of Theorem 6.2, is neither in $\text{LINEAR}_{\text{CS}}$ nor in LINEAR. Hence, the families $\text{LINEAR}_{\text{CS}}$ and LINEAR are not comparable to the family of real-time definable languages.

A characterization of $\text{LINEAR}_{\text{CS}}$ in terms of languages accepted by some class of machines has not been obtained. Since $\{wcw^Rcw \mid w \in \{a, b\}^*\}$ is not in $\text{LINEAR}_{\text{CS}}$, any such characterization could not depend on machines with the power to "remember and compare." However such machines would be able to "count," since in [1] the following theorem is established.

THEOREM 7.3. *The family of languages accepted by nondeterministic on-line multicounter machines which accept in linear time is a proper subfamily of* $\text{LINEAR}_{\text{CS}}$.

We now consider more general time bounds.

THEOREM 7.4. *Let f be the function $f(x) = 2^x$. Then $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ is the family of all* CS *languages.*

Proof. Recall that for any nondeterministic linear-bounded automaton M, there is a constant k such that for any input w, M either halts in an accepting state, halts in a nonaccepting state, or enters a loop within $k^{|w|}$ steps. Thus M accepts within a time bound $g(x) = k^x$. By Proposition 4.3, $L(M) \in \mathscr{L}_{\text{CS}}(g) \subseteq \mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ so that every CS language is in $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$. But $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ is a family of CS languages. Thus $\mathscr{S}(\mathscr{L}_{\text{CS}}(f))$ is the family of all CS languages.

PROPOSITION 7.5. *The family of languages accepted by nondeterministic multitape Turing acceptors which accept in polynomial time is precisely the family of languages generated by arbitrary grammars within polynomial time.*

Proof. This follows immediately from Proposition 4.5 and the results of [13] on the imitation of multitape machines by single tape machines.

It is an open question whether the languages of Proposition 7.5 are CS. In fact it is not known whether $\mathscr{L}(f)$ is a family of CS languages when $f(x) = x^2$.

Based on results in [3] and [13] and on Proposition 4.5, it is straightforward to show that if f is a bounding function such that $\inf_{x\to\infty} f(x)/x^2 > 0$, and g is a bounding function such that for all x, $g(x) > 2^{f(x)}$, then $\mathscr{L}(f) \subsetneqq \mathscr{L}(g)$ and $\mathscr{S}(\mathscr{L}(f)) \subsetneqq \mathscr{S}(\mathscr{L}(g))$. Hence there is an infinite hierarchy of families $\mathscr{L}(f) \subsetneqq \mathscr{L}(f') \subsetneqq \mathscr{L}(f'') \subsetneqq \cdots$ and an infinite hierarchy of AFLs $\mathscr{S}(\mathscr{L}(f)) \subsetneqq \mathscr{S}(\mathscr{L}(f')) \subsetneqq \mathscr{S}(\mathscr{L}(f'')) \subsetneqq \cdots$. It is not known if smaller differences in bounding functions yield hierarchies.

Acknowledgment

It is a pleasure to thank Professor Sheila A. Greibach for her many valuable suggestions and criticisms of this research, and Professor Susumu Kuno for his encouragement and support.

References

1. R. Book, "Grammars with Time Functions," Ph.D. dissertation, Harvard University, Cambridge, Mass., 1969; also appears as "Mathematical Linguistics and Automatic Translation, NSF-23," Aiken Computation Laboratory, Harvard University, Cambridge, Mass., 1969.
2. R. Book and S. Greibach, Quasi-realtime languages, *Math. Systems Theory* **4** (1970), 97–111.
3. R. Book, S. Greibach, and B. Wegbreit, Time- and tape-bounded Turing acceptors and AFLs, *J. Comput. System Sciences* **4** (1970), 606–621.
4. S. Ginsburg, "The Mathematical Theory of Context-Free Languages," McGraw-Hill, New York, 1966.
5. S. Ginsburg and S. Greibach, Mappings which preserve context-sensitive languages, *Information and Control* **9** (1966), 563–582.
6. S. Ginsburg and S. Greibach, "Abstract Families of Languages," Memoir No. 87, Amer. Math. Soc., Providence, R. I., 1969, 1–32.
7. A. Gladkii, On the complexity of derivations in phrase-structure grammars, *Algebri i Logika Sem.* **3** (1964), 29–44 (in Russian).
8. S. Greibach, "Inverses of Phrase Structure Grammars," Ph.D. dissertation, Harvard University, Cambridge, Mass., 1963; also appears as "Mathematical Linguistics and Automatic Translation, NSF-11," Aiken Computation Laboratory, Harvard University, Cambridge, Mass., 1963.
9. S. Greibach, A new normal-form theorem for context-free phrase-structure grammars, *J. Assoc. Comput. Mach* **12** (1965), 42–52.
10. S. Greibach, A note on undecidable problems of formal languages, *Math. Systems Theory* **2** (1968), 1–6.
11. S. Greibach and J. Hopcroft, "Independence of AFL Operations," Memoir No. 87, Amer. Math. Soc., Providence, R. I., 1969, 33–40.
12. T. Griffiths, Some remarks on derivations in general rewriting systems, *Information and Control* **12** (1968), 27–54.
13. J. Hartmanis and R. Stearns, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* **117** (1965), 285–306.

14. F. Hennie, One-tape off-line Turing machine computations, *Information and Control* **8** (1965), 553–578.
15. J. Hopcroft and J. Ullman, "Formal Languages and their Relation to Automata," Addison-Wesley, Reading, Mass., 1969.
16. S.-Y. Kuroda, Classes of languages and linear-bounded automata, *Information and Control* **7** (1964), 207–223.
17. P. Landweber, Three theorems on phrase-structure grammars, *Information and Control* **6** (1963), 131–136.
18. A. Paul, Properties of context-sensitive grammars, unpublished paper, Harvard University, Cambridge, Mass., 1967.
19. A. Rosenberg, Real-time definable languages, *J. Assoc. Comput. Mach.* **14** (1967), 645–662.