SIAM J. COMPUT.
Vol. 2, No. 4, December 1973

THE HARDEST CONTEXT-FREE LANGUAGE*

SHEILA A. GREIBACH†

Abstract. There is a context-free language L_0 such that every context-free language is an inverse homomorphic image of L_0 or $L_0 - \{e\}$. Hence the time complexity of recognition of L_0 is the least upper bound for time complexity of recognition of context-free languages. A similar result holds for quasirealtime Turing machine languages. Several languages are given such that deterministic and nondeterministic polynomial time acceptance are equivalent if and only if any one of them is deterministic polynomial time acceptable.

Key words. context-free, quasirealtime, complexity, polynomial time recognition

1. Introduction. There have been several studies of the complexity of recognition of context-free languages. It is known that the family of deterministic context-free languages is recognizable in linear time [1] but not in real time [2]. Context-free languages are recognizable off-line in space $\log^2 n$ [3]. They can be recognized by deterministic multitape Turing machines in time complexity n^3 [4]; linear context-free languages can be recognized in time n^2 [5].

It is not known if any of these upper bounds is also a lower bound. For example, it is not known whether there is any context-free language which cannot be accepted in linear time by a deterministic multitape Turing machine.

In this paper we show that there exists a "hardest" context-free language L_0 in the sense that any reasonable deterministic recognition procedure for the class of context-free languages will have as its time complexity the time it takes to recognize L_0 and will have as its tape complexity the space it takes to recognize L_0. The demonstration of this "universal" property of L_0 is constructive in the sense that an algorithm for recognizing L_0 in time $p(n)$ (p a polynomial or other semihomogeneous function[1]) or in space $p(n)$ can be mechanically transformed into an equally efficient recognizer for any other context-free language.

The idea behind this result can be extended to exhibit a specific language $\hat{L}_0$ holding the same relationships to $\mathcal{Q}$ (the family of languages accepted in "realtime" by nondeterministic multitape Turing machines). That is, if $\hat{L}_0$ can be accepted in time $p(n)$ by a deterministic multitape Turing machine for any polynomial p, then all members of $\mathcal{Q}$ can be so recognized. Since Book has shown that the class $\mathcal{P}$ of languages accepted by deterministic Turing machines in polynomial time is equal to $\mathcal{NP}$ (the class of languages accepted in polynomial time by nondeterministic Turing machines) if and only if $\mathcal{Q} \subseteq \mathcal{P}$ [6], it follows that $\mathcal{P} = \mathcal{NP}$ if and only if $\hat{L}_0$ is in $\mathcal{P}$. Furthermore, all reductions involved can actually be performed by polynomially time bounded Turing machines.

* Received by the editors December 1, 1972, and in revised form, April 18, 1973. The research represented in this paper was supported in part by the National Science Foundation under Grant no. GJ-803.

† Department of System Science, University of California at Los Angeles, Los Angeles, California 90024.

[1] A function f is *semihomogeneous* if for every $k_1 > 0$, there is a $k_2 > 0$, such that for all $x > 0$, $f(k_1x) \leqq k_2 f(x)$.

We show that every context-free language can be expressed[2] as $h^{-1}(L_0)$ or $h^{-1}(L_0 - \{e\})$ for a homomorphism h.[3] The algebraic statement is: the family of context-free languages is a principal AFDL; similarly, $\mathscr{Q}$ is a principal AFDL. By way of contrast, the family of deterministic context-free languages is not a principal AFDL [7].

2. Main results. In this section we exhibit a context-free language L_0 such that every context-free language is an inverse homomorphic image of L_0 or $L_0 - \{e\}$. As one might guess, L_0 is a nondeterministic version of a Dyck set.

DEFINITION 2.1. Let $T = \{a_1, a_2, \bar{a}_1, \bar{a}_2, c, ¢\}$ where $a_1, a_2, \bar{a}_1, \bar{a}_2, c$, and ¢ are all distinct. Let D be the context-free language generated by $S \to SS$, $S \to a_1 S \bar{a}_1$, $S \to a_2 S \bar{a}_2$, $S \to e$.[4] We call D a *Dyck set on two letters.* Let d be new and let

$$L_0 = \{e\} \cup \{x_1 c y_1 c z_1 d \cdots d x_n c y_n c z_n d \mid n \geqq 1, y_1 \cdots y_n \in ¢D, x_i, z_i \in T^* \text{ for all } i,$$
$$y_i \in \{a_1, a_2, \bar{a}_1, \bar{a}_2\}^* \text{ for } i \geqq 2\}$$

Thus the language L_0 selects one subword from each group set off by d's in such a way that the concatenation of the choices belongs to $¢D$, the Dyck set on two letters, preceded by ¢. In the construction below, we let L_0 encode derivations in a context-free grammar.

THEOREM 2.1. *If L is a context-free language, there is a homomorphism h such that* $L - \{e\} = h^{-1}(L_0 - \{e\})$.[5]

Proof. We may assume that L does not contain the empty word. Hence there is a grammar $G = (V, \Sigma, P, S)$ such that $L = L(G)$ and $P \subseteq [(V - \Sigma) \times \Sigma(V - \Sigma - \{S\})^*]$. Thus the rules in P are of the form $Z \to ay$ for $\bar{a} \in \Sigma$ and y containing neither terminals nor S; we say that G is in *standard* form [8].

Order $V - \Sigma$ in some way, $V - \Sigma = \{Y_1, \cdots, Y_n\}$, such that $Y_1 = S$. Define functions ξ, $\bar{\xi}$ from productions into $\{a_1, a_2, \bar{a}_1, \bar{a}_2\}^*$ as follows. If p is the production $Y_i \to a$, then $\bar{\xi}(p) = \bar{a}_1 \bar{a}_2^i \bar{a}_1$; if p is the production $Y_i \to a Y_{j_1} \cdots Y_{j_m}$, $m \geqq 1$, then $\bar{\xi}(p) = \bar{a}_1 \bar{a}_2^i \bar{a}_1 a_1 a_2^{j_m} a_1 \cdots a_1 a_2^{j_1} a_1$. If $i \neq 1$, $\xi(p) = \bar{\xi}(p)$; if $i = 1$, $\xi(p) = ¢ a_1 a_2 a_1 \bar{\xi}(p)$.

If $P_a = \{p_1, \cdots, p_m\}$ is the set of all productions whose right-hand sides start with a, then $h(a) = c\xi(p_1)c \cdots c\xi(p_m)cd$; without loss of generality we can assume $P_a \neq \varnothing$ for all a in Σ.

Thus we must show that $h(w) = x_1 c y_1 c z_1 d \cdots d x_k c y_k c z_k d$ with $k = |w|$,[6] $y_i \in \{a_1, a_2, \bar{a}_1, \bar{a}_2\}^*$, $x_i z_i \in T^*$, $1 \leqq i \leqq k$, and $y_1 \cdots y_k \in ¢D$ if and only if

[2] The symbol e is used for the empty string; L^+ is the closure of L under concatenation and $L^* = L^+ \cup \{e\}$.

[3] Notice that for any language L and homomorphism h, if L can be recognized within time $p(n)$ (tape $p(n)$), then $h^{-1}(L)$ can be so recognized, providing p is semihomogeneous [2], [16]; for $p(n) = 2^n$, e.g., this may not be true.

[4] A *context-free grammar* $G = (V, \Sigma, P, S)$ has finite vocabulary V, terminal vocabulary $\Sigma \subseteq V$, initial symbol $S \in V - \Sigma$ and finite production set $P \subset (V - \Sigma) \times V^*$. If $(Z, y) \in P$, written $Z \to y$, and $u, v \in V^*$, then $uZv \Rightarrow uyv$; the relation $\overset{*}{\Rightarrow}$ is the transitive reflexive extension of $\Rightarrow$. The *context-free language* generated by G is $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$.

[5] For a homomorphism h, $h^{-1}(L) = \{w \mid h(w) \in L\}$.

[6] For a word w, $|w|$ is the length of w.

$y_1 \cdots y_k$ encodes the productions used in a left-to-right derivation of w from S.[7]

It is well known (see [9] and [10] for further discussion) that D is the set of all words canceling to e under the cancellation rules $xa_1\bar{a}_1y \simeq e \simeq xa_2\bar{a}_2y$ and that for each w in $\{¢, a_1, a_2, \bar{a}_1, \bar{a}_2\}^*$ there is a unique word of minimal length, which we shall call $\mu(w)$, such that $w \simeq \mu(w)$. Further, w is in $\text{Init}\, D = \{x \mid \exists y, xy \in D\}$ if and only if $\mu(w) \in \{a_1, a_2\}^*$. It suffices to show that for $b_1, \cdots, b_k \in \Sigma$, $Y_{i_1}, \cdots, Y_{i_r} \in (V - \Sigma - \{S\})$, $S \overset{*}{\Rightarrow} b_1 \cdots b_k Y_{i_1} \cdots Y_{i_r}$ if and only if $h(b_1 \cdots b_k) = x_1cy_1czd \cdots dx_kcy_kcz_kd$, where $y_1 \cdots y_k \in \text{Init}\, ¢D$ and $\mu(y_1 \cdots y_k) = ¢a_1a_2^{i_r}a_1 \cdots a_1a_2^{i_1}a_1$. If $p_1, \cdots, p_k$ are the productions used, in order, in a left-to-right derivation of $b_1 \cdots b_k Y_{i_1} \cdots Y_{i_r}$, then $y_i = \xi(p_i)$, $1 \leqq i \leqq k$.

The proof is by induction on k; we shall sketch the induction step and leave other details to the reader. Assume the result for $k \geqq 1$. First, suppose $k \geqq 1$, $S \overset{*}{\Rightarrow} b_1 \cdots b_k Y_{i_1} \cdots Y_{i_r}$, production p is $Y_{i_1} \to b_{k+1} Y_{j_1} \cdots Y_{j_t}$ and $h(b_1 \cdots b_k) = x_1cy_1cz_1d \cdots dx_kcy_kcz_kd$, where $\mu(y_1 \cdots y_k) = ¢a_1a_2^{i_r}a_1 \cdots a_1a_2^{i_1}a_1$.

If we let

$$y_{k+1} = \xi(p) = \bar{a}_1\bar{a}_2^{i_1}\bar{a}_1a_1a_2^{j_t}a_1 \cdots a_1a_2^{j_1}a_1,$$

then $h(b_{k+1}) = x_{k+1}cy_{k+1}cz_{k+1}cd$, and

$$\begin{aligned}\mu(y_1 \cdots y_ky_{k+1}) &= ¢a_1a_2^{i_r}a_1 \cdots a_1a_2^{i_1}a_1y_{k+1} \\ &= a_1a_2^{i_r}a_1 \cdots a_1a_2^{i_2}a_1a_1a_2^{j_t}a_1 \cdots a_1a_2^{j_1}a_1,\end{aligned}$$

while

$$S \overset{*}{\Rightarrow} b_1 \cdots b_kY_{i_1} \cdots Y_{i_r} \Rightarrow b_1 \cdots b_kb_{k+1}Y_{j_1} \cdots Y_{j_t}Y_{i_2} \cdots Y_{i_r}.$$

On the other hand, suppose $h(b_1 \cdots b_{k+1}) = x_1cy_1cz_1d \cdots dx_{k+1}cy_{k+1}cz_{k+1}d$ and $\mu(y_1 \cdots y_{k+1}) \in ¢\{a_1, a_2\}^*$. Then, examining h, we see that we must have

$$\mu(y_1 \cdots y_k) = ¢a_1a_2^{i_r}a_1 \cdots a_1a_2^{i_1}a_1$$

and

$$\mu(y_{k+1}) = \xi(p) = \bar{a}_1\bar{a}_2^{s}\bar{a}_1a_1a_2^{j_t}a_1 \cdots a_1a_2^{j_1}a_1$$

for some production

$$p: Y_s \to b_{k+1}Y_{j_1} \cdots Y_{j_t}.$$

But $\mu(y_1 \cdots y_ky_{k+1}) \in ¢\{a_1, a_2\}^*$ implies that $s = i_1$. By the induction hypothesis,

$$S \overset{*}{\Rightarrow} b_1 \cdots b_kY_{i_1} \cdots Y_{i_r},$$

and hence

$$S \overset{*}{\Rightarrow} b_1 \cdots b_{k+1}Y_{j_1} \cdots Y_{j_t}Y_{i_1} \cdots Y_{i_r}.$$

We notice in passing that if we use the proof of Theorem 2.1 to express a context-free language L as $h^{-1}(L_0)$, the homomorphism h applied to a word w in

[7] If in every step of a derivation the leftmost nonterminal is expanded, it is a left-to-right derivation.

L encodes possible derivations for w in a context-free grammar G for L. Thus, for example, it is possible to select a grammar G_0 for L_0 such that the number of derivations of $h(w)$ in G_0 is precisely the number of derivations of w in G; in this sense homomorphism h "preserves multiplicities".[8] Furthermore, G_0 can be so selected that an efficient parser for L_0 (i.e., a recognizer which gives as output for y appropriate representations of derivations of y in G_0) can be converted automatically into an efficient parser for L.

DEFINITION 2.2. An AFDL is a family of languages containing at least one nonempty e-free[9] language and closed under inverse gsm mappings,[10] marked union and star,[11] and removal of endmarkers.[12] The least AFDL containing a language L is denoted by $\mathscr{F}_d(L)$ and is called a *principal* AFDL.

Since a homomorphism is a gsm mapping, the following corollary is immediate.

COROLLARY. *The family of context-free languages forms a principal* AFDL.

By the results of Chandler [11], the family of context-free languages can be accepted deterministically by a family of one-way machines with a *finite* number of tape symbols and instructions which define only context-free languages. So there is a deterministic acceptance scheme for the context-free languages. However, this scheme is of no practical use since it consists of reading an input, performing a finite state translation (a gsm mapping) onto a working tape, and then consulting an oracle at the end.

What is more significant is that we have a rather precise way of saying that the complexity of the family of context-free languages is precisely the complexity of the language L_0. If we have a deterministic machine M accepting a language L in time $p(n)$ and a homomorphism h, then we can construct from M and h an acceptor M' for $h^{-1}(L)$. Basically M' translates its input w symbol by symbol into $h(w)$ and feeds $h(w)$ into M. Thus w will be accepted by M' in time $p(|h(w)|) + |w|$ if $h(w)$ is accepted by M'. Hence if p is semihomogeneous (and polynomials are semihomogeneous) and M is a multitape Turing machine, we can build M' to accept $h^{-1}(L)$ in time $p(n)$. Similarly, if M is an off-line Turing machine accepting L in space $p(n)$, we can construct an off-line Turing machine accepting $h^{-1}(L)$ in space $p(n)$. (Notice that this argument applies not just to Turing machines but also to any "reasonable" deterministic recognition procedure.)

Hence, since we already know that L_0 as a context-free language *is* in $\mathscr{P}$, Theorem 2.1 tells us that the time necessary to recognize deterministically L_0 is

[8] The question of preservation of multiplicities was brought to the author's attention by S. Eilenberg.

[9] A language is *e-free* if it does not contain the empty word.

[10] A *gsm* is a sextuple $M = (K, \Sigma, \Delta, \delta, \lambda, q_0)$ where K, Σ, and Δ are finite sets of *states*, *inputs* and *outputs*, δ, the *transition* function, is a function from $K \times \Sigma$ into K, λ, the *output* function, is a function from $K \times \Sigma$ into Δ^* and $q_0 \in K$. We extend δ and λ to $K \times \Sigma^*$ by $\delta(q, e) = q$, $\lambda(q, e) = e$, $\delta(q, xa) = \delta(\delta(q, x), a)$ and $\lambda(q, xa) = \lambda(q, x)\lambda(\delta(q, x), a)$, for $q \in K$, $x \in \Sigma^*$ and $a \in \Sigma$. If $\delta(q, a) = p$, $\lambda(q, a) = y$, we sometimes write $(q, a) \rightarrow (p, y)$ and if $\delta(q, w) = p$, $\lambda(q, w) = y$, we write $(q, w) \overset{*}{\Rightarrow} (p, y)$ for $p, q \in K$, $a \in \Sigma$, $w \in \Sigma^*$ and $y \in \Delta^*$. We define $M(w) = \lambda(q_0, w)$, $M^{-1}(w) = \{y | M(y) = w\}$, $M(L) = \{M(w) | w \in L\}$ and $M^{-1}(L) = \{y | M(y)\} \in L$ for a word w or a language L, and call $M(L)$ a *gsm mapping* and $M^{-1}(L)$ an *inverse gsm mapping* of L.

[11] If $L_1 \cup L_2 \subseteq \Sigma^*$ and $c \notin \Sigma$, then $L_1 \cup cL_2$ is a *marked union* of L_1 and L_2 and $(L_1c)^*$ a *marked star* of L_1.

[12] If $c \notin \Sigma$ and $L \subseteq \Sigma^*c$, then c is an *endmarker* of L, and removing c yields $L/c = \{w | wc \in L\}$.

the (realizable) least upper bound of time complexity for deterministic recognition of the class of context-free languages. We believe L_0 to be the first known instance of such a language.

We can extend our arguments to show that the family $\mathcal{Q}$ is also a principal AFDL. The family $\mathcal{Q}$–called the family of *quasirealtime languages*–is the family of all and only those languages expressible as $h(L_1 \cap L_2 \cap L_3)$ where the L_i are context-free and h is a length preserving homomorphism[13][12]. For our purposes we can take this as a definition of $\mathcal{Q}$. Examining the proof of Theorem 2.1, we see that a similar construction would apply to members of $\mathcal{Q}$.

Instead of one Dyck set we need three. Let D' and D'' be two distinct renamings of D,[14] using vocabularies Σ_2 and Σ_3, where $D \subseteq \Sigma_1^*$.

We define the language $\hat{L}_0$ in words. It contains the empty string and every nonempty word in $\hat{L}_0$ ends with d's. Let $T = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \{c, ¢, d, \$\}$. A word w in T^*d containing m d's is in $\hat{L}_0$ if before the first d and then between each consecutive pair of d's in $\hat{L}_0$ one can select, sequentially, a triple of distinct nonempty subwords (x_i, y_i, z_i) such that

(i) $x_1 \cdots x_m \in ¢D$, $y_1 \cdots y_m \in ¢D'$ and $z_1 \cdots z_m \in ¢D''$, and

(ii) for each i, a subword of the form $\$\alpha_i c x_i c \beta_i c y_i c \gamma_i c z_i c \eta_i \$$ lies between the $(i-1)$st and ith d's and $\alpha_i \beta_i \gamma_i \eta_i$ contains *no* occurrence of $\$$ or d.

Thus $\hat{L}_0$ resembles a shuffle of three copies of L_0 with the crucial restriction that each triple selected lie within the *same* pair of $\$$'s. Akin to Theorem 2.1, we have the following result.

THEOREM 2.2. *If* $L \subseteq \Sigma^*$ *is in* $\mathcal{Q}$, *there is a homomorphism* h *such that*

$$L - \{e\} = h^{-1}(\hat{L}_0 - \{e\}).$$

Proof. The proof follows the lines of the previous one. We have $L = (L(G_1) \cap L(G_2) \cap L(G_3))$, where $\lambda: \Delta^* \to \Sigma^*$ is a length-preserving homomorphism and each G_i is a context-free grammar in standard form. For each G_i we define a homomorphism h_i from Δ^* into Σ_i^* similar to the one in the proof of Theorem 2.1; thus $h_i(b) = c\xi_i(p_1)c \cdots c\xi_i(p_m)c$, where ξ_i is the appropriate function mapping productions of G_i into Σ_i and $\{p_1, \cdots, p_m\}$ is the set of all productions of G_i of the form $Z \to by$. Then if $\lambda^{-1}(a) = \{b | \lambda(b) = a\} = \{b_1, \cdots, b_t\}$, we let

$$h(a) = \$h_1(b_1)h_2(b_1)h_3(b_1)\$ \cdots \$h_1(b_t)h_2(b_t)h_3(b_t)\$d.$$

Then $L - \{e\} = h^{-1}(\hat{L}_0 - \{e\})$ as desired.

COROLLARY 1. *Every language in* $\mathcal{Q}$ *is accepted deterministically in polynomial time if and only if* $\hat{L}_0$ *is so accepted.*

Now R. Book [6] has shown that $\mathcal{P}$ (the family of languages accepted by deterministic Turing machines in polynomial time) is equal to $\mathcal{NP}$ (the family of languages accepted by nondeterministic Turing machines in polynomial time) if and only if $\mathcal{Q} \subseteq \mathcal{P}$. Hence we have the following corollary.

COROLLARY 2. $\mathcal{P} = \mathcal{NP}$ *if and only if* $\hat{L}_0 \in \mathcal{P}$

Thus the membership problem for $\hat{L}_0$ is polynomially complete in the sense of Karp [13]. The main difficulty in recognizing $\hat{L}_0$ is that a triple x_i, y_i, z_i can be

[13] That is, for any symbol a, $h(a)$ is also a symbol.

[14] A *renaming* is a length-preserving one-to-one homomorphism h: $\Sigma_1^* \to \Sigma_2^*$, where $\Sigma_1 \cap \Sigma_2 = \varnothing$; if $L \subseteq \Sigma_1^*$, then $h(L)$ is a *renaming* of L.

selected only if cx_ic, cy_ic and cz_ic lie between the same pair of $\$$'s: if we could drop this restriction, we would get a member of $\mathscr{P}$. But we cannot–the resulting language is in the intersection closure of the context-free languages which is an AFDL properly contained in $\mathscr{Q}$ [12].

Indeed, results in [14] show that every language in $\mathscr{NP}$ can be obtained by polynomially-bounded erasing from a language $L_1 \cap L_2$ for L_1 and L_2 linear context-free.[15] Let us define

$$L_a = \{w\$_1 w^R | w \in \{a, b\}^*\} \quad \text{and} \quad L_1 = \{w\$_2 w^R | w \in \{0, 1\}^*\}.$$

Instead of the three Dyck sets used to form $\hat{L}_0$, "paste together" pieces of L_a and L_1 to form $\hat{L}_0'$:

$$\begin{aligned}\hat{L}_0' = \{e\} \cup \{du_1 ¢ v_1 c\alpha_1 c w_1 c\beta_1 c x_1 ¢ z_1 d \cdots du_1 ¢ v_m c\alpha_m c w_m c\beta_m c x_m ¢ z_m d | m \geqq 1,\\ \alpha_1 \cdots \alpha_m \in L_a, \beta_1 \cdots \beta_m \in L_1\\ \text{for } i \geqq 1, v_i w_i x_i \in \{a, b, 0, 1, \$_1, \$_2, c\}^*, \alpha_i \neq e \neq \beta_i,\\ u_i z_i \in \{a, b, 0, 1, \$_1, \$_2, c, ¢\}^*\}.\end{aligned}$$

Then combining arguments in [6], [14] and this paper, we have the next corollary.

COROLLARY 3. *$\mathscr{P} = \mathscr{NP}$ if and only if $\hat{L}_0' \in \mathscr{P}$.*

3. Conclusions and open problems.

We have shown that the family of context-free languages is a principal AFDL, although the deterministic context-free languages do not form a principal AFDL [7].

We can show certain other nondeterministic families to be principal AFDL's. Wegbreit's results can be extended to show that the nondeterministic and deterministic context-sensitive languages form principal AFDL's [15]. Similarly, the recursively enumerable languages form a principal AFDL. The various nondeterministic tape and time bounded and deterministic tape bounded Turing machine families described in [16] and [17] are easily seen to form principal AFDL's.[16] However the same question–do they form a principal AFDL–is open in other cases such as the one-way stack [18] and checking automata languages [19], and for various subfamilies of the context-free languages such as the one-counter languages; it is likewise unknown if every linear context-free language can be obtained from one language by inverse gsm mappings and derivatives. We suspect that the answer is no in all the cases mentioned, though for varying reasons.

Of course, the major open questions posed by this work concern the actual time complexity of L_0 and $\hat{L}_0$. These appear as difficult as ever.

Acknowledgments. An earlier version of this paper and other material appears in [7]. The author would like to thank the referees for useful suggestions in preparing this revision.

[15] A homomorphism h is *polynomially bounded* on L if there is a polynomial f such that $|w| \leqq f(|h(w)|)$ for all $w \in L$.

[16] Indeed, in all these cases one can express the family as $\{h^{-1}(L), h^{-1}(L - \{e\}) | h \text{ a homomorphism}\}$ for some generator L.

REFERENCES

[1] S. GINSBURG AND S. A. GREIBACH, *Deterministic context-free languages*, Information and Control, 9 (1966), pp. 602–648.

[2] A. L. ROSENBERG, *Real-time definable languages*, J. Assoc. Comput. Mach., 14 (1967), pp. 645–662.

[3] P. M. LEWIS, R. E. STEARNS AND J. HARTMANIS, *Memory bounds for recognition of context-free and context-sensitive languages*, IEEE Conference Record on Switching Circuit Theory and Logical Design, Ann Arbor, Michigan, 1965, pp. 191–202.

[4] D. H. YOUNGER, *Recognition and parsing of context-free languages in time n^3*, Information and Control, 10 (1967), pp. 189–208.

[5] T. KASAMI, *A note on computing time for recognition of languages generated by linear grammars*, Ibid., 10 (1967), pp. 209–214.

[6] R. V. BOOK, *On languages accepted in polynomial time*, this Journal, 1 (1972), pp. 281–287.

[7] S. GREIBACH, *Jump pda's, deterministic context-free languages, principal AFDL's and polynomial time recognition*, Proc. of Fifth Annual ACM Symposium on Theory of Computing, Austin, Texas, 1973, pp. 20–28.

[8] ———, *A new normal form theorem for context-free phrase structure grammars*, J. Assoc. Comput. Mach., 12 (1965), pp. 42–52.

[9] N. CHOMSKY, *Context-free grammars and pushdown storage*, MIT Res. Lab. Electron Quarterly Progress Rep. 65, Cambridge, Mass., 1962.

[10] N. CHOMSKY AND M. P. SCHUTZENBERGER, *The algebraic theory of context-free languages*, Computer Programming and Formal Systems, P. Braffort and P. Hirschberg, eds., North-Holland, Amsterdam, 1963, pp. 115–161.

[11] W. J. CHANDLER, *Abstract families of deterministic languages*, Proc. First ACM Symposium on Theory of Computing, Marina del Rey, Calif., 1969, pp. 21–30.

[12] R. V. BOOK AND S. GREIBACH, *Quasi-realtime languages*, Math. Systems Theory, 4 (1970), pp. 97–111.

[13] R. M. KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, pp. 85–104.

[14] B. BAKER AND R. V. BOOK, *Reversal-bounded multipushdown machines*, Proc. 13th Annual IEEE Symposium on Switching and Automata Theory, College Park, Maryland, 1972, pp. 207–211.

[15] B. WEGBREIT, *A generator of context-sensitive languages*, J. Comput. System Sci., 3 (1969), pp. 456–461.

[16] R. V. BOOK, S. GREIBACH AND B. WEGBREIT, *Time- and tape-bounded turing acceptors and AFL's*, Ibid., 4 (1970), pp. 606–621.

[17] R. BOOK, S. GREIBACH, O. IBARRA AND B. WEGBREIT, *Tape-bounded turing acceptors and principal AFL's*, Ibid., 4 (1970), pp. 622–625.

[18] S. GINSBURG, S. GREIBACH AND M. A. HARRISON, *One-way stack automata*, J. Assoc. Comput. Mach., 14 (1967), pp. 389–418.

[19] S. GREIBACH, *Checking automata and one-way stack languages*, J. Comput. System Sci., 3 (1969), pp. 196–217.