Math. Systems Theory **28**, 199–213 (1995)

Mathematical Systems Theory

Pumping Lemmas for the Control Language Hierarchy*

M. A. Palis[1] and S. M. Shende[2]

[1] Department of Electrical and Computer Engineering,
New Jersey Institute of Technology,
Newark, NJ 07102, USA
palis@hertz.njit.edu

[2] Department of Computer Science and Engineering,
University of Nebraska, Lincoln, NE 68588, USA
sunil@calypso.unl.edu

Abstract. We investigate a progression of grammatically defined language families, the *control language hierarchy.* This hierarchy has been studied recently from the perspective of providing a linguistic framework for natural language syntax. We exhibit a progression of pumping lemmas, one for each family in the hierarchy, thereby showing that the hierarchy is strictly separable.

1. Introduction

A large body of research in computational linguistics is devoted to the characterization of syntactic phenomena in natural language by means of grammatically defined formalisms. Recent research has suggested that a hierarchy of such formalisms, the *linear control languages*, may have some bearing on the study of natural language syntax. Control languages subsume the class of context-free languages. Moreover, they admit efficient sequential polynomial time and parallel $\mathbf{NC}^2$ recognition algorithms whose running times depend polynomially on the sizes of the corresponding grammars. We are interested in these languages from a formal perspective; in particular, we demonstrate that at every level in the hierarchy, there is a pumping lemma for that level, thereby settling the hitherto open question of strict separation of the hierarchy in the affirmative.

* The research reported in this paper was conducted in part at the Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA 19104, USA, and was supported under ARO Grant DAA29-84-9-0027, NSF Grants MCS-8219116-CER, MCS-82-07294, DCR-84-10413 and MCS-83-05221, and DARPA Grant N00014-85-K-0018.

2. Control Languages

Along with the elegant characterization of derivation tree paths in a context-free derivation by Thatcher [13], several researchers have studied the consequences of limiting derivations of context-free grammars, for instance, indexed grammars [1], EOL-grammars [4], matrix grammars [5], [11], state grammars [7], programmed context-free grammars [10], control grammars [2], and controlled linear context-free grammars [8]. An excellent summary of properties of some of these formalisms appears in the book by Salomaa [12]. We investigate a particular variation of control grammars called *linear control grammars*, based on the idea of independently controlling derivation paths in context-free derivations [16], [17]. For the sake of brevity, we henceforth denote these grammars simply as control grammars and the languages they generate as control languages.

Consider a context-free grammar with labeled productions. Informally, we first restrict ourselves to a subset of the paths in any derivation tree for the grammar and associate strings of production labels with these paths in a uniform way. Secondly, we *a priori* specify a language (also called *the control set*) to which these strings must belong. In particular, the control set can be a language of arbitrary complexity, e.g., a context-free language.

We assume that the reader is familiar with context-free grammars and derivations; our notation is basically consistent with Harrison [3]. A standard context-free grammar is a quadruple (V_N, V_T, P, Z), where V_N and V_T are, respectively, finite sets of *nonterminals* and *terminals*, with $Z \in V_N$ being the *start symbol* of the grammar. The set of *grammar symbols*, $V_N \cup V_T$, is denoted by V. P is a finite set of context-free productions of the form $\bar{p} = X \to X_1 \cdots X_n$, where $X \in V_N$ and the right-hand side $X_1 \cdots X_n$ belongs to V^*. The empty string is denoted by ε.

The following definition of control grammars[1] is adapted from Weir [16], [17].

Definition 2.1. Let $\bar{G} = (V_N, V_T, P, Z)$ be a standard context-free grammar. Let V_L be a finite set of *production labels* and let *Label*: $P \to V_L$ be a one-to-one function, which assigns to every production from P a unique label from V_L. In addition, for every production with nonempty right-hand side,

$$\bar{p} = X \to X_1 \cdots X_n,$$

there is a unique integer i, $1 \leq i \leq n$, that identifies the symbol X_i on the right-hand side of $\bar{p}$ as being *distinguished*. For a production

$$\bar{p} = X \to \varepsilon,$$

we denote the symbol ε as being distinguished. For the sake of clarity, if $Label(\bar{p}) = l$, then we write the labeled, distinguished production p obtained from $\bar{p}$ as

$$p = l\colon\ X \to X_1 \cdots \check{X}_i \cdots X_n \qquad \text{if} \quad n \geq 1,$$

[1] This definition is a variation of a general formalism proposed by Ginsburg and Spanier [2].

or as

$$p = l\colon\ X \to \check{\varepsilon}.$$

$G = (V_N, V_T, V_L, Z, P, Label)$ is called a Labeled, Distinguished Context-Free Grammar (or LDCFG) over the *underlying* context-free grammar $\bar{G}$. Let $C \subseteq V_L^+$ be some language (not containing the empty string ε) over the alphabet of labels V_L. Then $\mathcal{G} = \{G, C\}$ is defined to be a *control grammar*. Every string in V_L^+ is referred to as a *control string* or a *control word*. We say that the LDCFG G is *controlled* by the control set C, or that C is the control set of the LDCFG G in grammar $\mathcal{G}$ (for an example of a control grammar, see Figure 1(a)).

Consider a control grammar $\mathcal{G} = \{G, C\}$ as described above in Definition 2.1. Let $\bar{G}$ be the underlying context-free grammar of G. Following standard terminology, we say that $A \overset{*}{\underset{\bar{G}}{\Rightarrow}} \alpha$ if there is a standard context-free derivation in $\bar{G}$ of $\alpha \in V^*$ from $A \in V$ in zero or more steps; each step in the derivation corresponds to the context-free rewriting of some nonterminal symbol using an appropriate production of $\bar{G}$.

Then $A \overset{*}{\underset{G}{\Rightarrow}} \alpha$ (read A derives α in G) simply if $A \overset{*}{\underset{\bar{G}}{\Rightarrow}} \alpha$, i.e., a derivation of α from A in $\bar{G}$ is also a derivation in G. A derivation tree of the control grammar G is obtained by taking a derivation tree in the underlying context-free grammar $\bar{G}$ and decorating it as follows. For every internal node labeled X with its children labeled $X_1, \ldots, X_n$, we label the *edge* between the parent node (labeled X) and the child node (labeled X_i) with the production label l where $p = l\colon X \to X_1 \cdots \check{X}_i \cdots X_n$ is the labeled, distinguished production of G which corresponds to the production used to derive $X_1 \cdots X_n$ from X in the tree. We denote by $TreeSet(A \overset{*}{\underset{G}{\Rightarrow}} \alpha)$, the set of all such (decorated) derivation trees which correspond to the derivation $A \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

Figure 1(b) depicts a derivation tree in $TreeSet(Z \overset{*}{\underset{G}{\Rightarrow}} aabbcc)$ for the grammar $\mathcal{G}$ in Figure 1(a), with labeled edges shown in boldface. Note that from every node in

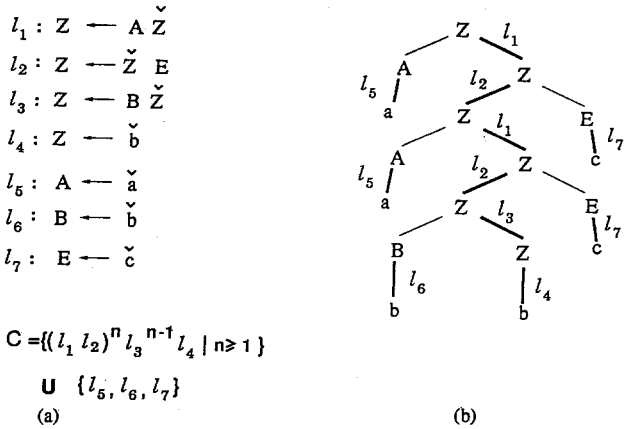

Fig. 1. A control grammar with a sample derivation tree.

the tree, there is a unique, edge-labeled path to some leaf node in the tree. In the rest of the paper an edge-labeled path will sometimes be identified with the control string labeling the path, i.e., the sequence of labels from the node beginning the path to the leaf node terminating the path. Given a derivation tree $\Gamma \in \mathit{TreeSet}(X \overset{*}{\underset{G}{\Rightarrow}} \alpha)$, the *maximal* labeled paths in Γ are called *c-paths*. Thus, a c-path ends at a leaf node and begins at some node which is either the root or an internal node which is connected to its parent by an unlabeled edge. Finally, we define $\mathit{ControlWords}(\Gamma)$ as the set of control strings that label c-paths in Γ. For example, for the derivation tree Γ shown in Figure 1(b), the set $\mathit{ControlWords}(\Gamma)$ is given by

$$\{l_1 l_2 l_1 l_2 l_3 l_4,\ l_5,\ l_6,\ l_7\}.$$

Definition 2.2. The Control Language $L(\mathscr{G})$, generated by the control grammar $\mathscr{G} = \{G, C\}$, with Z being the start symbol of G, is defined as

$$L(\mathscr{G}) = \left\{ w \in V_T^* \middle|\ \text{there is a derivation tree } \Gamma \in \mathit{TreeSet}\left(Z \overset{*}{\underset{G}{\Rightarrow}} w\right) \text{ such that } \mathit{ControlWords}(\Gamma) \subseteq C \right\}.$$

Let $\mathscr{C}$ be any family of languages over a finite alphabet. We say that a language L is *controlled in family* $\mathscr{C}$ if and only if there is a control grammar $\mathscr{G} = \{G, C\}$ such that $L = L(\mathscr{G})$ and $C \in \mathscr{C}$.

The reader may verify that the control language generated by the grammar in Figure 1(a) is the context-sensitive language $L(\mathscr{G}) = \{a^n b^n c^n | n \geq 1\}$, with the context-free control set C. Hence, $L(\mathscr{G})$ is controlled in the family of context-free languages, **CFL**, but is itself not a context-free language. Following Weir [16], a countable hierarchy of language families may be defined such that the zeroth level family in the hierarchy is the family of context-free languages, and, for every integer $i \geq 0$, a language in the $(i + 1)$th level family is generated by a control grammar whose control set is a language in the ith level family.

Definition 2.3. The Control Language Hierarchy (**CLH**) is defined as follows:

- $\mathbf{CLH}_0 = \mathbf{CFL}$, the family of context-free languages.
- For all $k \geq 1$,

  $\mathbf{CLH}_k = \{L |$ a context-free grammar G_0, and a sequence of LDCFGs $G_1, G_2, \ldots, G_k$ exist such that
  (1) $C_0 = L(G_0)$,
  (2) for all $1 \leq j < k$, $C_j = L(\{G_j, C_{j-1}\})$, and
  (3) $L = L(\{G_k, C_{k-1}\})\}$.

  We say that G_0 and the sequence of LDCFGs $G_1, G_2, \ldots, G_k$ *define* L.
- $\mathbf{CLH} = \{L | L \in \mathbf{CzLH}_k$ for some $k \geq 0\}$.

Languages in the hierarchy have interesting formal properties. For example, the languages generated at level one in the hierarchy are weakly equivalent to those

generated by *tree adjoining grammars*, a tree-rewriting formalism for natural language syntax [6], [14]–[16]. Every family in the hierarchy forms a full Abstract Family of Languages [17] and admits fast sequential and parallel parsing algorithms [9].

It is well known that the language $\{a^n b^n : n \geq 0\}$ is context free. This can be generalized by showing that, for any $i \geq 0$ and a finite alphabet $\{a_1, a_2, \ldots, a_{2^{i+1}}\}$, the language $LC_i = \{a_1^n a_2^n \cdots a_{2^{i+1}}^n : n \geq 0\}$ belongs to the family $\mathbf{CLH}_i$. For completeness, we provide a brief sketch of the proof. Let $k \geq 0$, and let $\mathscr{A} = \{a_1, a_2, \ldots, a_{2^{k+1}}\}$ be an alphabet. Assume inductively that $LC_k \in \mathbf{CLH}_k$ (over the alphabet $\mathscr{A}$). Let $a_0 \notin \mathscr{A}$ be a new symbol, and consider an alphabet $\mathscr{B} = \{b_1, b_2, \ldots, b_{2^{k+2}}\}$ disjoint from $\mathscr{A} \cup \{a_0\}$. We define an LDCFG, G_{k+1}, with control set $C = LC_k \cdot \{a_0\}$ such that $LC_{k+1} = L(\{G_{k+1}, C\})$; the set C belongs to $\mathbf{CLH}_k$ since the latter is an AFL. The grammar G_{k+1} has a single nonterminal Z and contains the production, a_0: $Z \rightarrow \check{\epsilon}$, along with 2^{k+1} other productions. In particular, for every j, $1 \leq j \leq 2^{k+1}$, the production,

$$a_j\colon\ Z \rightarrow b_j \check{Z} b_{2^{k+2}-j+1},$$

is in the grammar. It can be verified that the grammar $\{G_{k+1}, C\}$ generates exactly the strings in LC_{k+1}, over the alphabet $\mathscr{B}$.

Observe that every family $\mathbf{CLH}_i$, $i \geq 1$, can be individually separated from $\mathbf{CLH}_0$ by applying Ogden's pumping lemma for context-free languages [3] to the language $LC_i \in \mathbf{CLH}_i$. However, one of the questions that had remained open so far concerned the strict separation of the hierarchy, i.e., whether $\mathbf{CLH}_i \subset \mathbf{CLH}_{i+1}$, for all $i \geq 1$. We answer the question in the affirmative by exhibiting a progression of pumping lemmas, one for each level in the hierarchy.

3. Pumping Lemmas for the Hierarchy

For any $k \geq 0$, the family $\mathbf{CLH}_k$ is clearly contained in the family $\mathbf{CLH}_{k+1}$. We show that this containment is *proper* by proving a progression of pumping lemmas for families in the hierarchy. Before doing so, we note that Khabbaz [8] defined a subhierarchy contained in **CLH**, where the grammars G_i, $1 \leq i \leq k$, defining any given language at level k are restricted to be *linear* context-free grammars. He also established a progression of pumping lemmas thereby demonstrating strict separation of the Khabbaz subhierarchy. To our knowledge, it has not been formally shown that his hierarchy is a proper subhierarchy within **CLH**; nevertheless, our result is more general and provides a pumping lemma progression for the entire hierarchy **CLH**.

A control grammar, $\{G, C\}$, is said to be *reduced* if and only if every control word in C labels some c-path in a derivation tree of the grammar. Given a grammar $\mathscr{H} = \{H, D\}$ for L, we can easily construct an equivalent, reduced grammar $\mathscr{G} = \{G, C\}$ for L. Let M_H be the deterministic finite-state automaton corresponding to LDCFG H as follows. For every grammar symbol X of H (including the empty string ε), there is a state q_X in M_H. For every production l: $X \rightarrow X_1 \cdots \check{X}_i \cdots X_n$ of

H, there is a transition from state q_X to state q_{X_i} labeled l. All states of M_H corresponding to nonterminal symbols of H are *initial* states of M_H; the remaining states are designated as the *final* states. It is clear that the grammar $\mathcal{G} = \{G, C\}$, with $G = H$ and $C = D \cap L(M_H)$, also generates the language $L = L(\mathcal{H})$. Furthermore, the construction guarantees that the grammar $\mathcal{G}$ is a reduced control grammar for L, since only those control strings that label c-paths in partial derivation trees are retained in the final control set C. $\mathbf{CLH}_k$, for any $k \geq 0$, is a full AFL; consequently, we obtain the following result:

Proposition 3.1. *Let $k \geq 1$. For any language $L \in \mathbf{CLH}_k$, there is a grammar sequence $G_0, G_1, \ldots, G_k$ that defines L, such that if $L_0 = L(G_0)$ and L_i, $1 \leq i \leq k$, denotes the language defined by the subsequence of grammars $G_0, G_1, \ldots, G_i$, then $\{G_i, L_{i-1}\}$ is a reduced control grammar for L_i. In particular, $L_k = L$ is generated by the reduced grammar $\{G_k, L_{k-1}\}$.*

To simplify the statement of the pumping lemma and the subsequent proof, we establish the following notation. Consider an arbitrary alphabet, Σ, consisting of finitely many terminal symbols. Given a string $w = a_1 a_2 \cdots a_n$ with $a_i \in \Sigma$ $(1 \leq i \leq n)$, the *length*, n, of w is denoted as $|w|$. Every integer i, $1 \leq i \leq n$, is called a *position* of w. Informally, the position i refers to the ith symbol in w. Hence, specifying a subset of positions can be equivalently described as *marking* the corresponding symbols of w. For $j \geq 2$, a tuple of strings (over Σ), $\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_j)$, is called a *$j$-factorization* of the string $w_\Phi = \varphi_1 \varphi_2 \cdots \varphi_j$ obtained by concatenating the component strings in the tuple Φ. Conversely, for a given string w, Φ is a j-factorization of w if $w_\Phi = w$. For any $m \geq 0$, the string w^m is the concatenation of m identical copies of w (with w^0 denoting the empty string, ε).

Let F be some subset of positions in w. Then, any j-factorization, Φ, of w induces a partition of F given by the tuple $(F_1, F_2, \ldots, F_j)$; the component F_i, $1 \leq i \leq j$, contains exactly those positions in F which mark symbols of w in the substring φ_i in the factorization Φ. Formally, if $\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_j)$, then, for $1 \leq i \leq j$:

$$F_i = \{m \in F: \ |\varphi_1 \varphi_2 \cdots \varphi_{i-1}| < m \leq |\varphi_1 \varphi_2 \cdots \varphi_i|\}.$$

Next, we define the integer sequence e_i, $i \geq 0$, with geometric growth given by $e_i = 2^{(i+2)} + 1$. Note that $e_0 = 5$, $e_1 = 9$, and, in general, $e_{i+1} = 2e_i - 1$ for $i \geq 0$. For any given $i \geq 0$, let $\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_{e_i})$ be an e_i-factorization of the string w. Then the string $w_\Phi^{[m]}$ is defined by the e_i-factorization

$$\Phi^{[m]} = (u_1, u_2, \ldots, u_{e_i})$$

such that for any j, $1 \leq j \leq e_i$, the string $u_j = \varphi_j$ if j is odd, and $u_j = \varphi_j^m$ otherwise. For example, for $i = 1$, if the string $w = aba^2ba^2b^2a^2b^4$ has the e_1-factorization $\Phi = (ab, a, aba, ab, b, a^2, b^2, b, b)$, then the string $w_{\Phi^{[2]}} = aba^3ba^2bab^2a^4b^5$ is obtained from its e_1-factorization $\Phi^{[2]} = (ab, a^2, aba, (ab)^2, b, a^4, b^2, b^2, b)$. Note that, for any $i \geq 0$ and any factorization $\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_{e_i})$ of string w, exactly $(e_i - 1)/2 = 2^{i+1}$ substrings of w are pumped to yield the factorization $w_\Phi^{[m]}$.

For the sake of clarity, we adopt the following conventions. Derivation trees are named by the Greek letters Γ, Δ, etc. The letters Φ and Π name factorizations of strings; the corresponding lowercase letters with suitable subscripts are used to denote the component substrings of a factorization. We use lowercase Greek letters to denote nodes in derivation trees. Strings are generally identified by lowercase letters u, v, etc.

Theorem 1 (Pumping Lemma Progression). *Let $k \geq 0$. For every $L \in \mathbf{CLH}_k$, there is a constant n such that, for every $w \in L$ and any set of positions F in w, if $|F| \geq n$, then there is an e_k-factorization $\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_{e_k})$ of w that satisfies the following three statements. Let $(F_1, F_2, \ldots, F_{e_k})$ be the partition of F induced by the factorization Φ.*

(1) *There is a j, $1 \leq j \leq 2^{k+1}$, such that all the components of the triple $(F_{2j-1}, F_{2j}, F_{2j+1})$ are nonempty.*
(2) $|F_{2^{k+1}} \cup F_{2^{k+1}+1} \cup F_{2^{k+1}+2}| \leq n$.
(3) *For all $m \geq 0$, the string $w_\Phi^{[m]}$ also belongs to L.*

Observe that Theorem 1 for the case $k = 0$ is simply a restatement of Ogden's pumping lemma for CFLs [3]. In particular, for every context-free language L, there is a pumping constant n_0 such that, for every string w in the language with at least $|F| \geq n_0$ marked positions, there is a factorization

$$\Phi = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5)$$

of w satisfying the three statements in Theorem 1. The three statements in the theorem assert that:

1. Every component subset in the triple (F_1, F_2, F_3) or in the triple (F_3, F_4, F_5) is nonempty,
2. $|F_2 \cup F_3 \cup F_4| \leq n_0$.
3. For all $m \geq 0$, the string $\varphi_1 \varphi_2^m \varphi_3 \varphi_4^m \varphi_5 \in L$.

Theorem 1 is proved, in Section 3.2, by induction on $k \geq 0$, using Ogden's lemma as the basis for the argument. For $k \geq 1$, by Proposition 3.1, we can assume without loss of generality that the language $L \in \mathbf{CLH}_k$ is defined by the sequence of grammars $G_0, G_1, \ldots, G_k$, such that L is generated by the reduced control grammar $\mathcal{G} = \{G_k, C\}$ where the control set C is defined by the subsequence of grammars $G_0, G_1, \ldots, G_{k-1}$. The constant n in the statement of Theorem 1 is a property of the grammar sequence G_i, $0 \leq i \leq k$, and hence a property of the language L. To illustrate the techniques used in our proof, we describe the proof in detail for the case $k = 1$. The proof can be extended to higher levels in the hierarchy in a straightforward manner; we specify an appropriate constant $n = n(\mathcal{G})$.

Theorem 1 has a simple application: the separation of the control language hierarchy. For every integer i, $i \geq 0$, consider the finite alphabet $\Sigma_i = \{a_1, a_2, \ldots, a_{2^{i+1}}\}$. We remarked earlier that the language LC_i, $i \geq 0$, over the alphabet Σ_i defined as

$$LC_i = \{a_1^n a_2^n \cdots a_{2^{i+1}}^n : n \geq 0\},$$

belongs to the family $\mathbf{CLH}_i$.

Theorem 2. *For every integer i, $i \geq 1$, the language LC_i does not belong to the family* $\mathbf{CLH}_{i-1}$. *Hence, the hierarchy* **CLH** *is strictly separable.*

Proof. Fix $i \geq 1$ and assume for the sake of contradiction that LC_i belongs to the family $\mathbf{CLH}_{i-1}$. Let n be the pumping lemma constant for LC_i using $k = i - 1$ in the statement of Theorem 1. Consider the string $w = a_1^n a_2^n \cdots a_{2^{i+1}}^n$ in LC_i with *every* position in w being marked. By assumption, w satisfies the conditions in Theorem 1 for $k = i - 1$. Hence, there is an e_{i-1}-factorization

$$\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_{e_{i-1}})$$

of w for which statements (1)–(3) hold. However, statement (3) implies that we can obtain longer strings in the language LC_i by pumping exactly 2^i substrings of w given by the even-indexed components of the factorization Φ. Among the pumped substrings, at least one substring must be nonempty and to avoid scrambling symbols, each such nonempty substring must be of the form a_j^+ for some j, $1 \leq j \leq 2^{i+1}$. This leads to an immediate contradiction, since pumping only 2^i such substrings of w can never produce another string in LC_i. □

3.1. *Pumping Subtrees*

Before we turn to the proof of Theorem 1, we discuss the pumping of *subtrees* of a derivation tree of a control grammar. Given the language L, let $\{G, C\}$ be a reduced control grammar for L, with Z as the start symbol of LDCFG G and C as the control set. For a terminal string $w \in L$, let Γ in $\mathit{TreeSet}(Z \overset{*}{\underset{G}{\Rightarrow}} w)$ be a derivation tree for w. Consider a subtree Γ_0 of Γ with the property that the frontier of Γ_0 contains exactly one *internal node* of Γ. We denote this unique node of Γ as the *foot node* of Γ_0. The subtree Γ_0 is called an *auxiliary subtree* of Γ. An internal node in Γ is called a *source node* if it begins a c-path. An auxiliary subtree, Γ_0, is called a *recursive subtree* of Γ if both the root node and the foot node of Γ_0 are *source nodes* in Γ with the *same label.*

Under certain conditions, a sequence of auxiliary subtrees or a sequence of recursive subtrees contained in Γ, may be *pumped.* We say that any two auxiliary (or recursive) subtrees of Γ are disjoint if they do not share any node. For $r \geq 2$, consider a sequence $\Gamma_1, \Gamma_2, \ldots, \Gamma_r$ of mutually disjoint auxiliary subtrees of Γ such that there is a c-path, denoted P, from some source node of Γ to some leaf node of Γ, which passes through the root and foot nodes of each subtree Γ_j, $1 \leq j \leq r$. In particular, as shown in Figure 2(a), the path first passes through Γ_1, then through Γ_2, and so on, in sequence, finally passing through Γ_r. Let $\bar{u}$ be the control word labeling the c-path, and let $\Pi = (\pi_1, \pi_2, \ldots, \pi_{2r+1})$ be the factorization of $\bar{u}$ induced by the sequence of subtrees. Then the following result holds.

Proposition 3.2. *For every $m \geq 0$, if $\bar{u}_\Pi^{[m]} \in C$, then the tree Γ^m obtained from Γ by replacing each auxiliary subtree Γ_j, $1 \leq j \leq r$, by a stack of m* identical copies *of Γ_j (see Figure 2(b)), is also a derivation tree of the grammar. Further, if Γ derives a string in L, then so does Γ^m.*

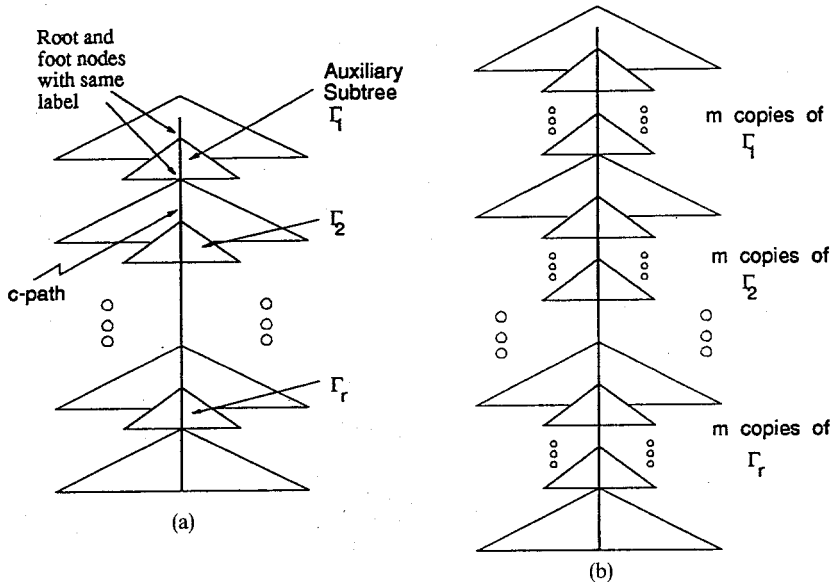

Fig. 2. Pumping auxiliary subtrees of Γ: (a) the tree Γ and (b) the tree Γ^m.

Proof. The first statement is a consequence of the fact that the control grammar is *reduced*. Specifically, if $\bar{u}_{\Pi}^{[m]} \in C$ for $m \geq 0$, then it follows from the reducedness of C that the root and foot nodes of each subtree Γ_j, $1 \leq j \leq r$, are labeled by the *same* nonterminal. Hence, the tree, Γ^m, is a well-defined derivation tree of the grammar.

If Γ derives a string in L, then every control word of Γ is in the set C. On the other hand, except for the path P, every other c-path in Γ either lies outside all the subtrees Γ_j, $1 \leq j \leq r$, or lies entirely inside some subtree Γ_j in the sequence. In the former case the c-path remains unchanged in Γ^m; in the latter case the c-path is replicated m times, once in each copy of the subtree Γ_j. It follows that every control word in Γ^m is also contained in the control set C, thus proving the second part of the proposition. □

A stronger result holds for a sequence of *recursive subtrees* in Γ. For $r \geq 2$, let $\Gamma_1, \Gamma_2, \ldots, \Gamma_r$ be a sequence of mutually disjoint recursive subtrees of a derivation tree Γ with the following property: there is a common root-to-leaf path in Γ which passes through the root and foot nodes of each recursive subtree Γ_j, $1 \leq j \leq r$, in that order (see Figure 3(a)).

Proposition 3.3. *For $m \geq 0$, the tree Γ^m obtained from Γ by replacing each recursive subtree Γ_j, $1 \leq j \leq r$, by a stack of m* identical copies *of Γ_j, is also a derivation tree of the grammar. Further, if Γ derives a string in L, then so does Γ^m.*

Proof. Recall that both the root and the foot nodes of every Γ_j, $1 \leq j \leq r$, are source nodes with the same nonterminal label in Γ. Hence, the first statement of the proposition is obvious.

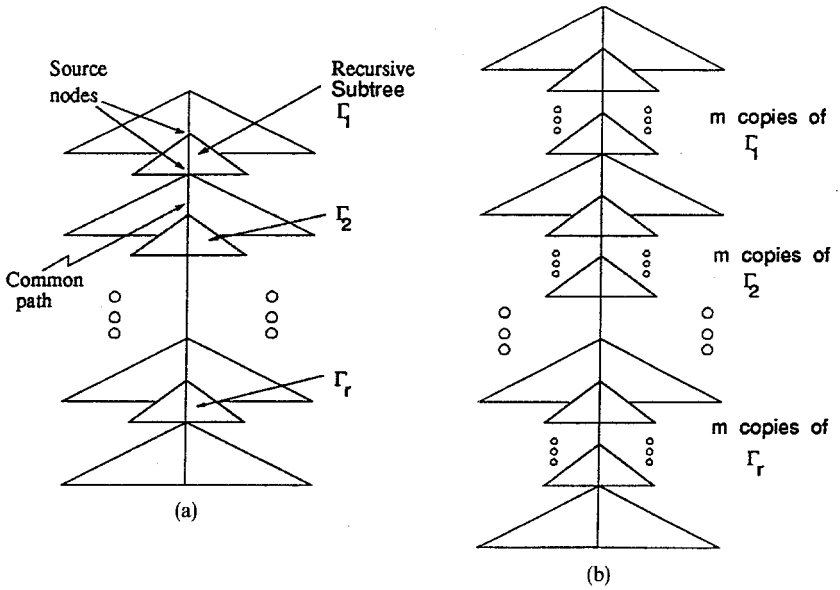

Fig. 3. Pumping recursive subtrees of Γ: (a) the tree Γ and (b) the tree Γ^m.

Γ has three kinds of c-paths:

(a) Paths that pass through nodes entirely outside the recursive subtree sequence.
(b) Paths that pass through nodes entirely inside some subtree Γ_j, $1 \leq j \leq r$.
(c) Paths that begin at the foot node of some subtree Γ_j, $1 \leq j \leq r$, but thereafter pass through nodes entirely outside the recursive subtree sequence.

In Γ^m, paths in categories (a) and (c) are unaffected, whereas each path in category (b) is simply replicated m times, once in each copy of the subtree (see Figure 3(b)). Clearly, the set of control words in Γ^m remains unchanged from those in Γ, proving the second statement in the proposition. □

Consider the sequence of subtrees $\Gamma_1, \Gamma_2, \ldots, \Gamma_r$ in Proposition 3.2 (or in Proposition 3.3). The sequence naturally induces a factorization

$$\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_{4r+1})$$

of the string w yielded by the derivation tree Γ. In particular, the substrings φ_{2j} and $\varphi_{4r-2j+2}$ are, respectively, the yields of the subtree Γ_j, $1 \leq j \leq r$, to the left and to the right of the foot node of Γ_j. It follows that if Γ derives the string $w \in L$, then the tree Γ^m derives the string $w_\Phi^{[m]} \in L$. Our proof of the pumping lemma progression relies crucially on this observation.

3.2. *The Proof*

We now proceed to establish Theorem 1 for the representative case $k = 1$, using Ogden's pumping lemma for context-free languages as the basis of the inductive

argument. Let $\mathcal{G}$ be a reduced control grammar for $L \in \mathbf{CLH}_l$ defined by the sequence of grammars G_0 and G_1, i.e., $\mathcal{G} = \{G_1, C\}$, where $C = L(G_0)$ is a context-free control set of the grammar. Let N_1 be the number of nonterminals of G_1, let d_1 be the maximum length of the right-hand sides of productions in G_1, and let n_0 be the context-free pumping lemma constant for G_0. Then we claim that the corresponding constant $n_1 \equiv n(\mathcal{G})$ is given by $n_1 = d_1^{4n_0(N_1+1)}$.

Consider a string $w \in L(\mathcal{G})$ and let F be a set of positions of w with $|F| \geq n_1$. Let Γ be a derivation tree for w, with $ControlWords(\Gamma) \in C$, where the leaf nodes corresponding to positions in F are *marked*. We define some subsets of the set of nodes of Γ with the following simplifying notation: for every such subset, say A, the nodes contained in set A are called A-nodes.

A D-node[2] in Γ is an *ancestor* of some marked leaf corresponding to a position in F. A B-node is a D-node with the following additional property: it has at least *two* immediate children in Γ which are both D-nodes. In other words, there is a path to some marked leaf in Γ from a D-node, whereas a B-node is one from which there are at least two disjoint paths to marked leaves. The following result may be easily established (see p. 187 of [3]).

Proposition 3.4. *Let Γ' be any subtree of Γ with the property that, for some integer $i \geq 0$, every root-to-leaf path in Γ' has at most i B-nodes. Then Γ' has at most d_1^i marked leaves corresponding to positions in F.*

By Proposition 3.4, a root-to-leaf path in Γ with at least $4n_0(N_1+1)$ B-nodes exists; let P be a path with the maximum number of B-nodes over all root-to-leaf paths in Γ. Let the path P end at the leaf node α; note that α may or may not be a marked leaf.

Denote by $\bar{P}$ the smallest suffix of the path P which contains the leaf node α, begins at a B-node, and contains exactly $4n_0(N_1 + 1)$ B-nodes, i.e., $\bar{P}$ contains the *lowest* $4n_0(N_1 + 1)$ B-nodes along path P (see Figure 4). Let $\bar{P}$ begin at the B-node, γ, and let Δ be the subtree of Γ rooted at node γ. From the definition of $\bar{P}$, it follows that every path in Δ has at most $4n_0(N_1 + 1)$ B-nodes. Hence, by Proposition 3.4, the number of marked leaves on the frontier of Δ is at most n_1 (see Figure 4).

Let $B_{\bar{P}}$ denote the set of B-nodes on path $\bar{P}$. Note that every $B_{\bar{P}}$-node is inside the subtree Δ. Now, every $B_{\bar{P}}$-node is an ancestor of some marked leaf on the frontier of the subtree Δ, such that the leaf either lies to the left or to the right of the leaf node α. Define B_l (B_r) to be the subset of $B_{\bar{P}}$, such that every B_l-node (resp. B_r-node) has a marked descendant to the left (resp. right) of the leaf node α. Clearly, $B_l \cup B_r = B_{\bar{P}}$, even though the sets B_l and B_r may not be disjoint. Since $B_{\bar{P}}$ contains exactly $4n_0(N_1 + 1)$ nodes, at least half of those nodes must belong either to B_l or to B_r. Without loss of generality, assume that B_l contains at least $2n_0(N_1 + 1)$ nodes.

Every node on path $\bar{P}$ is either a source node or is a descendant of a source node on path P. Let S be the set of source nodes on path $\bar{P}$, and let β be the unique source node defined as follows. If the root node, γ, of subtree Δ is an S-node, then $\beta = \gamma$.

[2] The notation is consistent with the terminology used on p. 187 of [3].

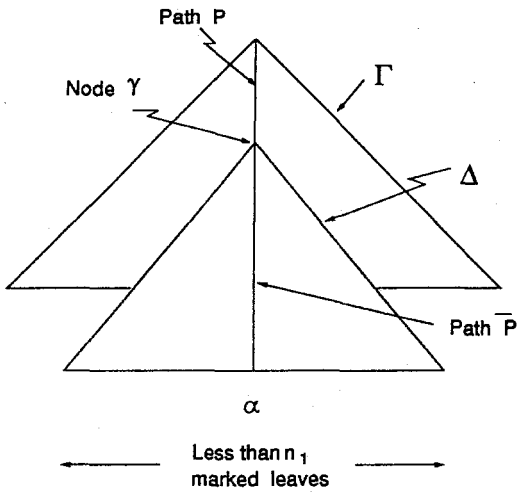

Fig. 4. The subtree Δ.

Otherwise, β is the minimal source node on path P which is an ancestor of γ, i.e., the c-path starting at β enters the subtree Δ at the node γ. Every B_1-node belongs to the c-path of some source node in the set $S \cup \{\beta\}$. We have two possible cases:

Case 1: There is a source node in $S \cup \{\beta\}$ such that the c-path starting at the node has at least n_0 B_1-nodes (on the path $\bar{P}$). Choose any such source node, and let $\bar{u}$ be the control word labeling its c-path. For every B_1-node encountered on the c-path, we *mark* the label on the labeled edge (along the c-path) below the node. These marked labels now serve as *positions* of the control word $\bar{u}$. We denote the set of these positions by K; by our choice of the source node, the cardinality of the set K is at least n_0. Hence, applying the inductive hypothesis to the control word $\bar{u}$, an e_0-factorization, $\Pi = (\pi_1, \ldots, \pi_{e_0})$, of $\bar{u}$ satisfying the conditions of Ogden's pumping lemma exists. Recall that $e_0 = 5$; accordingly, let K_i, $1 \leq i \leq 5$, be the subset of marked positions in the substring π_i, and consider the e_1-factorization

$$\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_{e_1})$$

of w induced by the factorization Π of the control word $\bar{u}$ (see Figure 5). Since $e_1 = 9$, let F_i, $1 \leq i \leq 9$, be the set of positions in the substring φ_i. We prove that the factorization, Φ, satisfies the three statements in the lemma.

1. By the inductive hypothesis, either the subsets K_1, K_2, and K_3 or the subsets K_3, K_4, and K_5 are nonempty. Since every marked position of the control word corresponds to a distinct B_1-node in subtree Δ, it follows that, for $1 \leq i \leq 5$, every marked position in K_i corresponds to a distinct marked position in F_i. From this, we conclude that either the subsets F_1, F_2, and F_3 or the subsets F_3, F_4, and F_5 are nonempty. This proves statement (1) of Theorem 1 (the other symmetric cases in the statement arise when there are more than $2n_0(N_1 + 1)$ B_r-nodes, and can be proved in a similar manner).

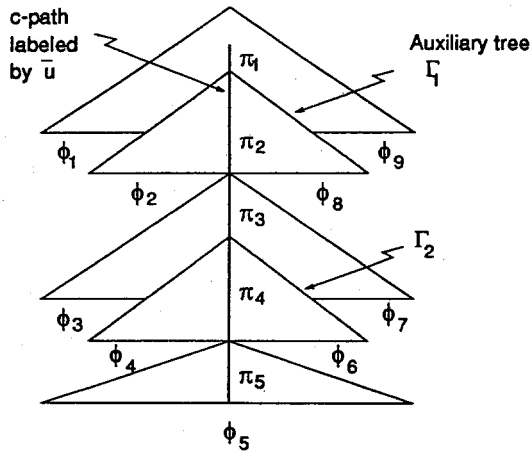

Fig. 5. Pumping a c-path with at least n_0 B_1-nodes.

2. From the observations in the preceding paragraph, the portion of the c-path, labeled by the suffix $\pi_4\pi_5$ of the control word $\bar{u}$, is entirely inside the subtree Δ. It follows that $\varphi_4\varphi_5\varphi_6$ is a substring of the yield of Δ. We conclude that $|F_4 \cup F_5 \cup F_6| \leq n_1$, proving statement (2) of the theorem. For the symmetric case when there are more B_r-nodes than B_l-nodes, a similar argument suffices.
3. By inductive hypothesis, $\bar{u}_\Pi^{[m]} \in C$ for all $m \geq 0$. However, the substrings π_2 and π_4 induce the sequence of *auxiliary subtrees* Γ_1 and Γ_2 with the root and foot nodes of both trees lying on the c-path, as shown in Figure 5. Hence, Proposition 3.2 applies with $r = 2$, and we conclude that $w_\Phi^{[m]} \in L$ for all $m \geq 0$. This proves statement (3) of the theorem.

Case 2: There are less than n_0 B_1-nodes on every c-path starting at a source node in $S \cup \{\beta\}$. Since there are at least $2n_0(N_1 + 1)$ B_1-nodes, there must be no less than $(2N_1 + 3)$ source nodes from the set $S \cup \{\beta\}$, such that their corresponding c-paths contain at least one B_1-node along $\bar{P}$. Of these, at least $2(N_1 + 1)$ nodes belong to the set S. Choose $2(N_1 + 1)$ such S-nodes, denoted by the ordered sequence $\alpha_1, \alpha_2, \ldots, \alpha_{2(N_1+1)}$, with the property that for every i, $1 \leq i \leq 2(N_1 + 1)$, the S-node α_i is an ancestor of the S-node α_{i+1} along path $\bar{P}$. Let the node α_i, $1 \leq i \leq 2(N_1 + 1)$, be labeled X_i.

There are only N_1 nonterminals in the grammar G_1 and $2(N_1 + 1)$ nonterminals labeling the sequence of nodes α_i, $1 \leq i \leq 2(N_1 + 1)$. Hence, by the pigeonhole principle, two pairs of nodes $(\alpha_{r_i}, \alpha_{f_i})$, $1 \leq i \leq 2$, exist such that $(i - 1)(N_1 + 1) < r_i < f_i \leq i(N_1 + 1)$ and $X_{r_i} = X_{f_i}$. In other words, Γ contains two disjoint *recursive subtrees,* Γ_1 and Γ_2, both sharing disjoint portions of the path P. It is readily seen that the path P is partitioned into five segments by the recursive subtrees; the segments, in turn, induce an e_1-factorization

$$\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_9)$$

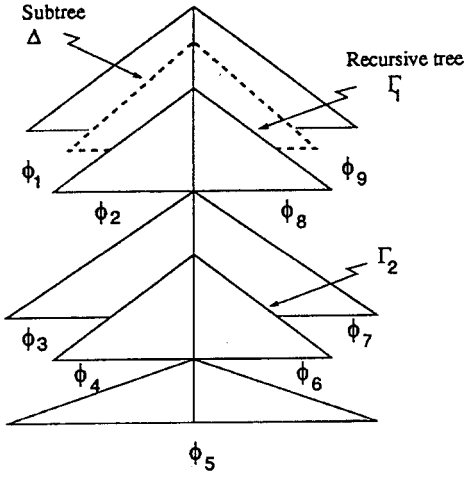

Fig. 6. Pumping a recursive subtree sequence inside Δ.

of w (see Figure 6). As before, let F_i, $1 \leq i \leq 9$, be the subset of positions of the substring φ_i in the factorization. We again prove that the factorization, Φ, satisfies the three statements in the theorem.

1. Statement (1) follows from the fact that there is at least one B_l-node in each of the last four segments of the path P, because this implies that there is a marked position in each of the substrings φ_2, φ_3, φ_4, and φ_5 of w. The symmetric cases in statement (1) arise when there are more B_r-nodes than B_l-nodes on the path $\bar{P}$, and can be proved in a similar manner.
2. Observe that the recursive subtree, Γ_1, is entirely inside the subtree Δ. Hence, it follows that $\varphi_2\varphi_3 \cdots \varphi_8$ is a substring of the yield of subtree Δ. Statement (2) is immediate; in fact, we have
$$|F_2 \cup F_3 \cup \cdots \cup F_8| \leq n_1.$$
3. Applying Proposition 3.3 (with $r = 2$) to the recursive tree sequence Γ_1 and Γ_2, we can conclude that the factorization, Φ, satisfies $w_\Phi^{[m]} \in L$ for all $m \geq 0$.

This completes the proof of the theorem for the case $k = 1$. In general, we can extend the argument by induction for levels $k > 1$ in a systematic manner. Let N_k be the number of nonterminals of G_k, and, inductively, let n_{k-1} be the pumping lemma constant for the control set C, defined by the sequence of grammars $G_0, G_1, \ldots, G_{k-1}$. Let d_k be the maximum length of the right-hand sides of the productions in grammar G_k, and define the pumping constant n_k for L as

$$n_k = d_k^{2^{k+1} n_{k-1}(N_k+1)}.$$

We first use Proposition 3.4 with the constant d_k. This provides a path, P, and a minimal suffix, $\bar{P}$, of the path P with exactly $2^{k+1} n_{k-1}(N_k + 1)B$-nodes on $\bar{P}$.

Following the same case analysis, in the first case we assume that there is a source node in the set, $S \cup \{\beta\}$, with at least n_{k-1} B_l-nodes (or B_r-nodes). The inductive hypothesis for level $(k-1)$ yields a sequence of $r = 2^k$ auxiliary subtrees, which can be pumped using Proposition 3.2 (with $r = 2^k$). This disposes of the first case. Otherwise, we obtain a sequence of $r = 2^k$ recursive subtrees, and pump the sequence using Proposition 3.3 (with $r = 2^k$). In both cases it is easy to verify that statements (1)–(3) in Theorem 1 are satisfied by the factorization induced by the corresponding sequence of subtrees. With these modifications, the theorem can be proved for any $k \geq 1$. □

Acknowledgments

We are grateful to the anonymous referees and, especially, the editor, Prof. Joost Engelfriet, for carefully reading the manuscript, pointing out some subtle errors, and for suggesting improvements to the overall presentation of the paper.

References

[1] A. V. Aho. Indexed grammars—an extension to context free grammars. *J. Assoc. Comput. Mech.*, 15:647–671, 1968.
[2] S. Ginsburg and E. H. Spanier, Control sets on grammars. *Math. Systems Theory*, 2:159–177, 1968.
[3] M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, MA, 1978.
[4] G. T. Herman and G. Rozenberg. *Developmental Systems and Languages.* North-Holland, Amsterdam, 1975.
[5] O. H. Ibarra. Simple matrix languages. *Inform. and Control*, 17:359–394, 1970.
[6] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *J. Comput. System Sci.*, 10(1), 1975.
[7] T. Kasai. An hierarchy between context-free and context-sensitive languages. *J. Comput. System Sci.*, 4:492–508, 1970.
[8] N. A. Khabbaz. A geometric hierarchy of languages. *J. Comput. System Sci.*, 8:142–157, 1974.
[9] M. A. Palis and S. Shende. Upper bounds on recognition of a hierarchy of non-context-free languages. *Theoret. Comput. Sci.*, 98(2):289–319, May 1992.
[10] D. J. Rozenkrantz. Programmed grammars and classes of formal languages. *J. Assoc. Comput. Mech.*, 16:107–131, 1969.
[11] A. Salomaa. Matrix grammars with a leftmost restriction. *Inform. and Control*, 20:143–149, 1970.
[12] A. Salomaa. *Formal Languages.* Academic Press, New York, 1973.
[13] J. W. Thatcher. Tree automata: an informal survey. In A. V. Aho, editor, *Currents in the Theory of Computing*, pp. 143–172. Prentice-Hall, Englewood Cliffs, NJ, 1973.
[14] K. Vijay-Shanker. A Study of Tree Adjoining Grammars. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1987.
[15] K. Vijay-Shanker and D. J. Weir. The equivalence of four extensions of context-free grammars. *Math. Systems Theory*, 27:511–546, 1994.
[16] D. J. Weir. Characterizing Mildly Context-Sensitive Grammar Formalisms. Ph.D. Thesis, University of Pennsylvania, Philadelphia, PA, September 1988.
[17] D. J. Weir. A geometric hierarchy beyond context-free languages. *Theoret. Comput. Sci.*, 104:235–261, 1992.

Received February 6, 1991, and in revised form June 15, 1992, and March 10, 1993, and in final form April 29, 1993.