October 1990

Descriptional Succinctness of Some Grammatical Formalisms for Natrual Language

Michael A. Palis
University of Pennsylvania

Sunil Shende
University of Nebraska

Descriptional Succinctness of Some Grammatical Formalisms for Natrual Language

Abstract

We investigate the problem of describing languages compactly in different grammatical formalisms for natural languages. In particular, the problem is studied from the point of view of some newly developed natural language formalisms like linear control grammars (LCGs) and tree adjoining grammars (TAGs); these formalisms not only generate non-context-free languages that capture a wide variety of syntactic phenomena found in natural language, but also have computationally efficient polynomial time recognition algorithms. We prove that the formalisms enjoy the property of unbounded succinctness over the family of context-grammars, i.e. they are, in general, able to provide more compact representations of natural languages as compared to standard context-free grammars.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-82.

Descriptional Succinctness of Some Grammatical Formalisms for Natural Language

MS-CIS-90-82
LINC LAB 187

Michael A. Palis
University of Pennsylvania

Sunil Shende
University of Nebraska

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

November 1990

Descriptional Succinctness of Some Grammatical Formalisms for Natural Language[*]

Michael A. Palis
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

and

Sunil Shende
Department of Computer Science
University of Nebraska
Lincoln, NE 68588-0115

October 25, 1990

[*]Supported by ARO grant DAA29-84-9-0027, NSF grants MCS-8219116-CER, MCS-82-07294, DCR-84-10413, MCS-83-05221 and DARPA grant N00014-85-K-0018.

Abstract

We investigate the problem of describing languages compactly in different grammatical formalisms for natural languages. In particular, the problem is studied from the point of view of some newly developed natural language formalisms like linear control grammars (LCGs) and tree adjoining grammars (TAGs); these formalisms not only generate non-context-free languages that capture a wide variety of syntactic phenomena found in natural language, but also have computationally efficient polynomial time recognition algorithms. We prove that the formalisms enjoy the property of unbounded succintness over the family of context-free grammars, i.e. they are, in general, able of provide more compact representations of natural languages as compared to standard context-free grammars.

1 Introduction

A significant body of research in computational linguistics is devoted to the characterization of syntactic phenomena in natural language by means of formalisms that can be processed efficiently. By now, a variety of grammatical formalisms, including context-free grammars, have been considered to be useful in specifying natural language to varying degrees of linguistic adequacy. However, not all of them have efficient recognition algorithms and among those that do, the effect of the size of the grammar on parsing complexity is still far from clear.

Among the different candidate formalisms, context-free grammars, by virtue of their simplicity of description and their ease of processing, are widely accepted as a reference against which other formalisms may be compared in terms of both linguistic power as well as computational efficiency. Recent research has shown that a family of formalisms, the *linear control languages*, not only give a more satisfactory linguistic account of natural language syntax but also possess fast sequential polynomial time and parallel $\mathbf{NC}^2$ recognition algorithms. The purpose of this note is to provide a different perspective on the comparison between context-free and linear control grammars, viz. that a grammar in the latter formalism can be unboundedly more *succinct* in terms of grammar size than one in the former for the same language.

2 Linear control languages and descriptive succinctness

Following the characterization of derivation tree paths in a context-free derivation by Thatcher [12], several researchers have studied the consequences of limiting derivations of context-free grammars, e.g. EOL-grammars [4], matrix grammars [11, 5], state grammars [7], programmed context-free grammars [10], and controlled linear context-free grammars [8].

Linear control grammars exploit the idea of controlling context-free derivations in a novel fashion. We first restrict ourselves to a subset of the paths in a derivation tree of the context-free grammar and associate strings of *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 [1]. 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 $\bar{p} = X \rightarrow X_1 \ldots X_n$, where $X \in V_N$ and the right-hand side $X_1 \ldots X_n$ belongs to V^*.

The following definition of control grammars is adapted from Weir [14].

$$\begin{array}{lll} l_1 : Z_1 & \rightarrow & A\check{Z}_1 \\ l_2 : Z_1 & \rightarrow & \check{Z}_1 E \\ l_3 : Z_1 & \rightarrow & B\check{Z}_1 \\ l_4 : Z_1 & \rightarrow & \check{b} \\ l_5 : A & \rightarrow & \check{a} \\ l_6 : B & \rightarrow & \check{b} \\ l_7 : E & \rightarrow & \check{c} \end{array} \qquad C \;=\; \begin{array}{l} \{(l_1 l_2)^n {l_3}^{n-1} l_4 \mid n \geq 1\} \\ \cup \; \{l_5, l_6, l_7\} \end{array}$$

LDCFG productions of G_1 Control set C

Figure 0.1: Control Grammar $\mathcal{G} \;=\; \{G_1, C\}$

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 *Label* : $P \rightarrow V_L$, a one-to-one function, which assigns to every production from P, a unique label from V_L. In addition, for every production

$$\bar{p} = X \rightarrow X_1 \ldots X_n$$

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 the sake of clarity, if *Label*$(\bar{p}) \;=\; l$, then we write the labeled, distinguished production p obtained from $\bar{p}$ as

$$p \;=\; l : X \rightarrow X_1 \ldots \check{X}_i \ldots X_n$$

$G \;=\; (V_N, V_T, V_L, Z, P, \mathit{Label})$ is called a Labeled, Distinguished Context-Free Grammar (or LD-CFG) over the *underlying* context-free grammar $\bar{G}$. Now, 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 *linear 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 $\mathcal{G}$.

An example of a control grammar is shown in figure 0.1.

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 $A \underset{\bar{G}}{\overset{*}{\Longrightarrow}} \alpha$ 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 $A \underset{G}{\overset{*}{\Longrightarrow}} \alpha$ (read A derives α in G) simply if $A \underset{\bar{G}}{\overset{*}{\Longrightarrow}} \alpha$, i.e. a derivation of α from A in $\bar{G}$ is also a derivation in G. A derivation tree of the linear control grammar G is simply obtained by taking a derivation tree in the underlying context-free grammar $\bar{G}$ and decorating it as follows.

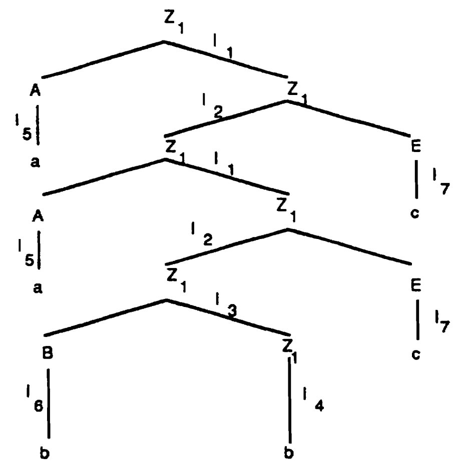

Figure 0.2: Derivation Tree associated with $Z_1 \underset{G_1}{\overset{*}{\Longrightarrow}} aabbcc$

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 $p = l : X \rightarrow X_1 \ldots \check{X}_i \ldots X_n$ is the labeled, distinguished production of G which corresponds to the production used to derive $X_1 \ldots X_n$ from X in the tree. We denote by $TreeSet(A \underset{G}{\overset{*}{\Longrightarrow}} \alpha)$, the set of all such (decorated) derivation trees which correspond to the derivation $A \underset{G}{\overset{*}{\Longrightarrow}} \alpha$.

Figure 0.2 shows a derivation tree in $TreeSet(Z_1 \underset{G_1}{\overset{*}{\Longrightarrow}} aabbcc)$ for the grammar $\mathcal{G}$ in Figure 0.1. Note that from every node in 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 to the leaf node. For simplicity, we shall also refer to the path as a labeled path or a *control* path.

Given a derivation tree $\Gamma \in TreeSet(X \underset{G}{\overset{*}{\Longrightarrow}} \alpha)$, we denote the unique control path from the root node to a leaf node of Γ by $Spine(\Gamma)$ (or simply, as the *spine* if Γ is clear from the context). Thus, $l_1 l_2 l_1 l_2 l_3 l_4$ is the spine in Figure 0.2. The (unique) leaf node which terminates $Spine(\Gamma)$ is called the *foot node* of Γ. Finally, $ControlWords(\Gamma)$ is the set of control strings which label all the *maximal* control paths (hereafter called *c-paths*) in Γ. A c-path ends at a leaf node and begins at some node which is either the root or a internal node which is connected to its parent by an *unlabeled edge.* In particular, note that $Spine(\Gamma) \in ControlWords(\Gamma)$.

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

$$L(\mathcal{G}) = \{a_1 \ldots a_n \in V_T{}^* \mid \text{ there is a derivation tree } \Gamma \in TreeSet(Z \underset{G}{\overset{*}{\Longrightarrow}} a_1 \ldots a_n), \text{ and } ControlWords(\Gamma) \subseteq C\}.$$

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

The reader may verify that the control language generated by the grammar in Figure 0.1 is the context-sensitive language $L(\mathcal{G}) = \{a^n b^n c^n \mid n \geq 1\}$, with the context-free control set C. $L(\mathcal{G})$ is, therefore, controlled in the family of context-free languages, **CFL**, but is itself not a context-free language.

Following Weir [14], a countable hierarchy of language classes may be defined such that the 0-th family in the hierarchy is exactly the family of context-free languages, and every language in the $(i+1)$-th family is generated by a control grammar whose control set is a language in the i-th family.

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

- $\mathbf{CLH_0} = \mathbf{CFL}$, the family of context-free languages.
- for all $k \geq 1$,
  $\mathbf{CLH_k} = \{L \mid$ there exists a context-free grammar G_0, and a sequence of LDCFGs $G_1, G_2, \ldots, G_k$ 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.
- $\mathbf{CLH} = \{L \mid L \in \mathbf{CLH_k}$ for some countable $k \geq 0\}$.

Languages in the hierarchy have very interesting properties from the point of view of computational linguistics. For example, the languages generated at level one in the hierarchy are basically identical to those generated by *tree adjoining grammars*, a formalism for natural language which not only has significant linguistic appeal but also has an $O(n^6)$ sequential parsing algorithm. In general, it was shown by Palis et.al. [9] that every level in the hierarchy admits fast sequential and parallel parsing algorithms. Moreover, all the levels are closed under linguistically plausible operations like concatenation. However, the complexity of all the parsing algorithms depends polynomially on the size of the grammar being parsed. From a formal standpoint, this is not a serious consideration but from the perspective of practical parsing, the question of succinct grammatical representation becomes very important and interesting.

Informally, we say that a grammar is more succinct than another if they both generate the same language, but the former is smaller in size than the latter. Comparing between two *different* formalisms is somewhat more involved. Given a specific class of languages, if there is a clear functional relationship between the sizes of the minimal representations from each class, then the function which maps the minimal size of one representation to the other determines, in some sense, how much economy one formalism possesses over the other. For example, it is well known that one can always construct, within an exponential bound, a deterministic finite automaton which is language equivalent to some given non-deterministic one. This can also be qualitatively expressed. We would say that non-deterministic automata are exponentially *more* succinct than deterministic automata.

Formally, let Σ and Γ be disjoint alphabets.

Definition 2.4 A formalism $\mathcal{F} = \{N_i \in \Gamma^* \mid i \geq 1\}$ is a countable set of strings (or representations) such that to every $N_i \in \mathcal{F}$, we can associate a language $L(N_i) \subseteq \Sigma^*$ in a uniform way. Let $[\mathcal{F}] = \{L(N_i) \mid i \geq 1\}$ denote the family of languages represented by $\mathcal{F}$.

Let the size of N_i be given by the number of symbols in N_i. Let $\mathcal{L}$ be a family of languages, and $\mathcal{F}_1$ and $\mathcal{F}_2$ be two given formalisms such that $\mathcal{L} \subseteq [\mathcal{F}_1]$ and $\mathcal{L} \subseteq [\mathcal{F}_2]$.

Definition 2.5 [3] The relative succinctness of $\mathcal{F}_2$ over $\mathcal{F}_1$ with respect to the family $\mathcal{L}$ is recursively bounded if and only if there is a recursive function f with the property that for any language $L \in \mathcal{L}$, its minimal size representations N_i and K_j, in $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively, satisfy $f(\mid K_j \mid) \geq \mid N_i \mid$.

We show next that no such recursive bound exists for, say, the class of context-free languages when the two representations being compared are context-free grammars and linear control grammars at level one in the hierarchy. This implies that the latter formalism must, in general, be more succinct than the former, and in fact, must be unboundedly more succinct.

3 Main Results

Consider a Turing machine M, and let $ID_0(x)$ be an instantaneous description of the machine (i.e. state, input head, and worktape head information) on some given input x. A halting computation of M on x (if it indeed halts) is then a finite sequence of instantaneous descriptions $ID_0(x)$, $ID_1(x)$, $\ldots$, $ID_j(x)$, where $ID_j(x)$ is a description of a halting configuration of M on x. Let $[z]^R$ denote the string obtained by reversing z. Consider the language $VALC(M)$ which consists of strings of the form

$$ID_0(x)\#[ID_1(x)]^R\#ID_2(x)\ldots ID_{2j}(x)$$

or of the form

$$ID_0(y)\#[ID_1(y)]^R\#ID_2(y)\ldots[ID_{2j+1}(y)]^R,$$

where both $ID_{2j}(x)$ and $ID_{2j+1}(y)$ are descriptions of halting configurations of M on x and y, respectively.

It is well known [6] that $VALC(M)$ can be described as the intersection of two context-free languages, and that $INVALC(M) = \Sigma^* - VALC(M)$ is, in fact, a context-free language generated by a grammar obtained effectively from a description of M. We can modify every machine M so that it always goes through an initial sequence of two states, before proceeding with the rest of its computation. Using this observation, we can derive the following important result.

Theorem 3.1 [6] Given a Turing machine M, $VALC(M)$ is context-free if and only if $L(M)$ is finite.

Clearly, if $L(M)$ is finite, it has only a finite number of halting computations and so $VALC(M)$ must be finite, and hence context-free. Conversely, if $L(M)$ is infinite, there is some long string x accepted by M such that the size of $ID_2(x)$ is longer than the constant used for Ogden's pumping lemma for context-free languages. It follows that pumping substrings within $ID_2(x)$ will result in strings which do not describe genuine halting computations of M. Hence, $VALC(M)$ cannot be context-free if $L(M)$ is infinite.

The proof outlined in the previous paragraph rests crucially on Ogden's pumping lemma. In the next section, we demonstrate a much stronger result, viz. that every level $i \geq 0$ family of linear control languages has an Ogden-style "pumping" lemma. As a consequence, a slight modification of the proof above allows us to obtain the following:

Theorem 3.2 For any $i \geq 0$, $VALC(M)$ is in CLH_i if and only if $L(M)$ is finite.

Many results about recursive succinctness appear in [13, 3, 2] etc. Most of these results are proved by using a variation on the basic argument[1] that we present below. We will show that for any $i \geq 1$, the relative succinctness between linear control grammars at level i and at level $(i-1)$ with respect to the family of *co-finite languages* is recursively unbounded.

For assume that such a recursive bound f exists. Then, given a grammar at level i, say G, of size n, we can recursively enumerate all the (finitely many) level $(i-1)$ grammars of size at most $f(n)$, and then perform the following procedure. For every such level $(i-1)$ grammar, G_j, we successively test each input string (in lexicographic order, say) on both G and G_j. If there is even one string which is not accepted by both machines, then the sub-computation associated with the pair (G, G_j) is terminated (note that the recognition problem for every level in the hierarchy is decidable; in fact, it is decidable in polynomial time).

Now, if all such sub-computations terminate, then we may conclude that there is no level $(i-1)$ grammar of size at most $f(n)$ which is equivalent to (i.e. recognizes the same language as) grammar

[1]Hartmanis demonstrates some very general conditions in [2] under which there can be no recursive succinctness bound between two formalisms. Our result can also be derived as a corollary from his general theorems.

K. By the definition of recursive succinctness, $L(K)$ must not be co-finite. Then, the foregoing procedure gives us a way to enumerate the set of all level i grammars which recognize languages that are not co-finite. However, by theorem 3.2, such an enumeration could be used to effectively enumerate all Turing machines M which accept infinite sets, by simply checking to see if a level i grammar for the language $INVALC(M)$ is ever enumerated. We get a contradiction, since the infiniteness (of languages accepted) property for Turing machines is not recursively enumerable.

Theorem 3.3 The relative succinctness between level i linear control grammars ($i \geq 1$) and level $(i-1)$ grammars with respect to the family of co-finite languages is recursively unbounded. Hence, the relative succinctness between the two formalisms with respect to the family CLH_{i-1} is also recursively unbounded.

4 Pumping lemma for the hierarchy

For any k, the family $\mathbf{CLH_k}$ is contained in the family $\mathbf{CLH_{k+1}}$; in this section, we show that this containment is *proper* by proving a pumping lemma scheme for the hierarchy. Khabbaz defined a special case of our hierarchy by using an identical construction, where the grammars G_i, $1 \leq i \leq k$, are restricted. In fact, a pumping lemma similar to ours is proved in [8] to demonstrate strict separation of his hierarchy. However, the scheme cannot be used for our purposes, since families at any level $k \geq 1$ in the Khabbaz hierarchy are *not closed under concatenation* unlike their counterparts $\mathbf{CLH_k}$. Consequently, at each level $k \geq 1$, the family $\mathbf{CLH_k}$ *properly contains* the corresponding Khabbaz family. [2]

Let G be an arbitrary LDCFG. Productions in G of the form $l : X \rightarrow \check{\epsilon}$ and $l : X \rightarrow \check{Y}$ for nonterminals X, Y are respectively called ϵ-productions and *chain*-productions of G. It can be shown that:

Lemma 4.1 For any $k \geq 0$, let $L = L(\{G, C\})$ be a control language in $\mathbf{CLH_{k+1}}$, where C is in $\mathbf{CLH_k}$. Then there is a control grammar $\mathcal{H} = \{H, D\}$ such that $L = L(\mathcal{H})$, D is in $\mathbf{CLH_k}$, and the underlying grammar of LDCFG H has no ϵ - or chain-productions.

Given a grammar $\mathcal{H} = \{H, D\}$ for L as above, we construct an equivalent grammar $\mathcal{G} = \{G, C\}$ for L. Let M_H be the deterministic finite-state automaton corresponding to LDCFG H as follows. For every grammar symbol X of H, there is a state q_X in M_H. For every production $l : X \rightarrow X_1 \ldots \check{X}_i \ldots X_n$ 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 should be easy to see that the grammar $\mathcal{G} = \{G, C\}$ with $G = H$

[2]For example, for every language L at level $k \geq 1$ in the Khabbaz hierarchy, the language $L\dot{L}$ obtained by concatenating L to itself, is in $\mathbf{CLH_k}$ but not in the corresponding Khabbaz family

and $C = D \cap L(M_H)$, also generates the language $L = L(\mathcal{H})$. We shall say that the grammar $\mathcal{G}$ is a *reduced* control grammar for L. Since $\mathbf{CLH_k}$ for arbitrary k is closed under intersection with regular languages, the following result is obvious.

Lemma 4.2 For any k and any language $L \in \mathbf{CLH_k}$, there is a grammar sequence $G_0, G_1, \ldots, G_k$ generating L with the properties that:

- $G_0 = (V_0, \Sigma_0, P_0, Z_0)$ is a standard context-free grammar free of ϵ - and chain-productions,
- for all $1 \leq i \leq k$, $G_i = (V_i, \Sigma_i, \Pi_i, Z_i, P_i, \mathit{Label}_i)$ is an LDCFG free of ϵ - and chain-productions,
- if, for all $0 \leq i \leq k$, L_i is the language generated by the grammar sequence $G_0, G_1, \ldots, G_i$, then $\{G_i, L_{i-1}\}$ is a reduced control grammar for L_i.

We define a j-factorization Φ of a string w to be a tuple of strings $(u_1, \ldots, u_j)$ such that $w = u_1 \ldots u_j$, i.e. w is obtained by concatenating components of its factorization. If the string w has length n, then every integer i, $1 \leq i \leq n$, is called a *position* of w. Informally, the position i refers to the i^{th} symbol in w. Hence, specifying a set of positions can also be described as *marking* the corresponding symbols of w.

Given a set of positions, F, in w, any j-factorization Φ of w naturally induces a partition of F given by $(F_1, F_2, \ldots, F_j)$, such that the component F_i, $1 \leq i \leq j$, contains the set of positions in F which mark symbols of w in the substring u_i in Φ. Formally, if $\Phi = (u_1, \ldots, u_j)$, then $F/\Phi = (F_1, \ldots, F_j)$ is defined by

$$F_i = \{m \in F \mid lg(u_1 \ldots u_{i-1}) < m \leq lg(u_1 \ldots u_i)\}, \quad 1 \leq i \leq j$$

For the sake of readability, we define the integer sequence e_i, $i \geq 0$, given by $e_i = 2^{(i+2)} + 1$. Note that $e_{i+1} = 2e_i - 1$.

Theorem 4.3 (Pumping Lemma Scheme) For any $k \geq 0$, let $L = L(\mathcal{G})$ be a language in $\mathbf{CLH_k}$ generated by the grammar sequence $\mathcal{G} = G_0, G_1, \ldots, G_k$ satisfying the conditions of Lemma 4.2.

Then there is a constant $n(\mathcal{G})$ such that for each $w \in L$, and any set of positions F in w, if $\mid F \mid \geq n(\mathcal{G})$ then there is an e_k- factorization $\Phi = (v_1, \ldots v_{e_k})$ of w with the property that

1. at least one triple $(F_{2j-1}, F_{2j}, F_{2j+1})$, for $1 \leq j \leq e_{k-1} - 1$, satisfies $F_p \neq \epsilon$, for $2j - 1 \leq p \leq 2j + 1$,
2. $\mid F_2 \cup F_3 \ldots \cup F_{e_k - 1} \mid \leq n(\mathcal{G}$, and
3. for all $m \geq 0$, the string $w^{[m]}$ with factorization

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

also belongs to L, where the strings u_i, $1 \leq i \leq e_k$ are defined by $u_i = v_i$ if i is odd, and $u_i = {v_i}^m$ otherwise.

Notice that Theorem 4.3 for $k = 0$ is simply a restatement of Ogden's strong pumping lemma for CFLs [1]. We prove the general result by induction, using the context-free pumping lemma as basis. To illustrate the technique, we shall first provide the proof for the case $k = 1$. Extensions to higher levels in the hierarchy are obtained by specifying an appropriate constant $n(\mathcal{G})$.

At the outset, we make two simple combinatorial observations.

Proposition 4.4 Let G be an arbitrary LDCFG with N distinct nonterminal symbols. Then given any $m \geq 1$, and a sequence of nonterminals $(X_0, X_1, \ldots, X_{mN})$ of G, there exist m ***distinct pairs*** of integers (p_i, q_i), $1 \leq i \leq m$, such that $(i-1)N \leq p_i < q_i \leq iN$ and $X_{p_i} = X_{q_i}$.

The proposition is easily proved by applying the pigeonhole principle m times to contiguous segments of length $(N+1)$ of the sequence, i.e. to the segment $(X_0, \ldots, X_N)$, the segment $(X_N, \ldots, X_{2N})$ etc.

Now let $\{G, C\}$ be a reduced control grammar with Z as the start symbol of LDCFG G, and consider some derivation tree Γ in $TreeSet(Z \overset{G}{\underset{*}{\Longrightarrow}} w)$ for some *terminal* string w. A node in Γ is called a *source node* if it begins a c-path.

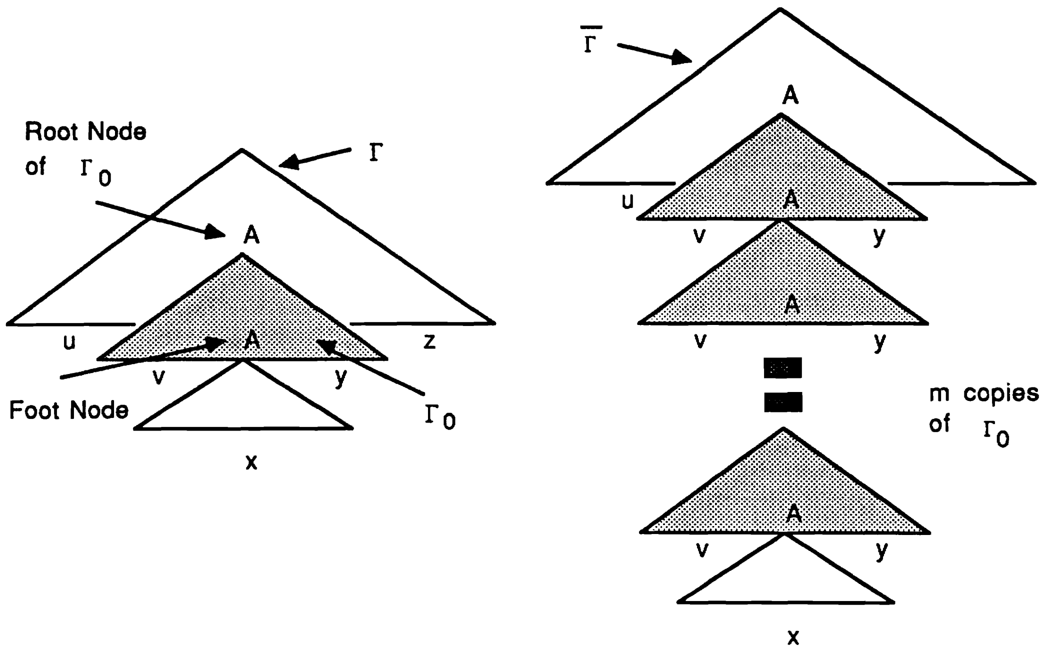

Figure 0.3: A Recursive Subtree of Γ

Consider the subtree Γ_0 of Γ as shown in Figure 0.3. Γ_0 is called a *recursive subtree* of Γ if and only if both its root node and the unique internal node of Γ (denoted as the *foot node* of Γ_0) which is at the frontier of Γ_0, are labeled by the same nonterminal symbol A. Furthermore, both the root node and the foot node of Γ_0 are required to be *source nodes* in Γ. As shown in the figure, the recursive subtree Γ_0 induces a factorization $\Phi = (u, v, x, y, z)$ of the string w. By Lemma 4.1, the string vy is always non-empty.

Proposition 4.5 Let Γ_0 be a recursive subtree of a valid derivation tree Γ for $w \in L(\{G, C\})$, i.e. $Words(\Gamma) \subseteq C$. Then the tree $\bar{\Gamma}$ obtained from Γ by replacing Γ_0 by a stack of $m \geq 0$ ***identical copies*** of Γ_0 (see Figure 0.3), is also a valid derivation tree for a string in $L(\{G, C\})$. In particular, if Γ derives the string $w = uvxyz$ as shown then $\bar{\Gamma}$ derives the string $uv^m xy^m z$.

Proof: (of Claim 4.5): Observe that since the root and the foot nodes of Γ_0 are also source nodes in Γ, every c-path of Γ either passes through nodes *entirely inside* Γ_0 or through nodes *entirely outside* Γ_0 (the c-path which begins at the foot node of Γ_0 belongs to the latter category). But $Words(\Gamma) \subseteq C$; hence, all control words which label c-paths in Γ_0 are all *in the control set C*. Therefore, replacing Γ_0 by a stack of m of its identical copies within Γ produces derivation tree $\bar{\Gamma}$ which is also valid, i.e. with $Words(\bar{\Gamma}) \subseteq C$, for the replacement simply produces copies of the c-paths already in Γ_0 without extending any of the c-paths originally in Γ. Note that if (u, v, x, y, z) is the factorization of w induced by Γ_0, then the substrings v and y on the frontier of Γ are replaced by v^m and y^m in $\bar{\Gamma}$ as shown in Figure 0.3. □

We now proceed to prove Theorem 4.3 for the case $k = 1$. Let $\mathcal{G} = \{G_0, G_1\}$ be a control grammar for $L \in CLH_1$ as in Lemma 4.2. Let N_1 be the number of nonterminals of 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 $n_1 = {d_1}^{2n_0(4N_1+3)}$ where d_1 is the maximum length of right hand sides of productions in G_1.

Some preliminary definitions and observations are needed next. Let w be a string in $L(\mathcal{G})$ and let F be a set of positions of w with $| F |> n_1$. Then for any derivation tree, Δ, for w, we can define the following subsets of the set of nodes of Δ. A $D-$node [3] is an *ancestor* of some position in F. A $B-$node is a $D-$node with the following additional property. It has at least *two* immediate sons in Γ which are both $D-$nodes. Stated somewhat differently, every node which is on the path from the root node to some position in F belongs to the set of $D-$nodes. The $B-$nodes are simply those which belong to at least two such distinct paths. Given these definitions, it is easy to show ([1], pp.187) that

Proposition 4.6 For every tree Γ, if w is the string of terminal symbols at the leaves of Γ, F is a set of marked positions of w, and every root-to-leaf path in Γ has at most i $B-$nodes, then the number of positions of w in F is at most ${d_1}^i$.

[3]The notation is consistent with the terminology used in [1], pp.187

Recall that we chose $\mid F \mid$ to be greater than n_1. Consequently, by the contrapositive of proposition 4.6, there is a path in Γ with at least $2n_0(4N_1+3)$ $B-$nodes; without loss of generality, let P be the path with the maximum number of $B-$nodes over all such paths. P begins at the root node of Γ and ends at some leaf node labeled, say, by the l^{th} symbol of w, denoted as w_l.

Consider, $\bar{P}$, which is the smallest contiguous part of the path P such that it contains the leaf node labeled w_l and has exactly $2n_0(4N_1+3)$ $B-$nodes, i.e. $\bar{P}$ starts at some $B-$node, denoted $\bar{\gamma}$ and contains the "lowest" $2n_0(4N_1+3)$ $B-$nodes of P (see Figure 0.4). We denote the set of

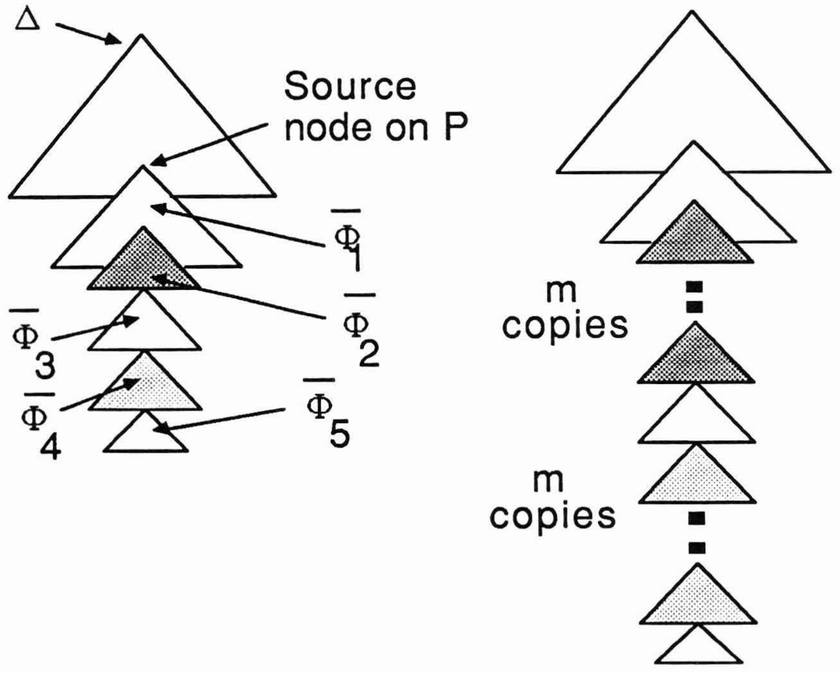

Figure 0.4: Pumping a c-path from a source node on P

$B-$nodes in $\bar{P}$ by $B_{\bar{P}}$, and the subtree of Γ rooted at $\bar{\gamma}$ by Δ. It is easy to see that every path in Δ has at most $2n_0(4N_1+3)$ $B-$nodes; hence, by proposition 4.6, the number of positions on the frontier of Δ is at most n_1. All the strings that will be "pumped" in the remainder of the proof belong to the frontier of Δ. This takes care of condition (2) in Theorem 4.3.

Let S be the set of source nodes on $\bar{P}$. For any $B-$node in S, it must be an ancestor of some position from F in w in subtree Γ, such that the position either lies to the left or to the right of the leaf node labeled w_l. We shall denote the set of $B-$nodes in S with marked descendents to the left of w_l as B_l; the set B_r is defined analogously. It is easy to see that $B_l \cup B_r = B_{\bar{P}}$. Since $B_{\bar{P}}$ contains exactly $2n_0(4N_1+3)$ nodes, at least half of those nodes must belong either to B_l or to B_r.

We now have three cases depending on the number of *source nodes* on path $\bar{P}$ (recall that source nodes begin c-paths in tree Γ); Either $\mid S \mid$ equals 0, or $\mid S \mid$ is between 1 and $(4N_1+3)$ (inclusive), or $\mid S \mid$ is at least $4(N_1+1)$.

1. $\mid S \mid = 0$: Since none of the internal nodes on $\bar{P}$ are source nodes, it follows from our definitions that $\bar{P}$ forms the "tail" of the c-path beginning at some source node on path P (see Figure 0.4).

   Without loss of generality, suppose that B_l contains at least half of the nodes in $B_{\bar{P}}$. For every node in B_l, we *mark* the label on the directed edge out of the node (note that every $B-$node has such a directed edge out of it which lies on $\bar{P}$). These marked labels now serve as "positions" on the control word. We denote the set of these positions by K; the size of K equals that of B_l, and is clearly greater than n_0. Hence, by Ogden's lemma (or Theorem 4.3 with $k = 0$), we can find a $5-$factorization, $\bar{\Phi}$, of the control word, such that either K_1, K_2, K_3 or K_3, K_4, K_5 are all non-empty with respect to $\bar{\Phi}$. But the factorization $\bar{\Phi}$ of the control word induces a $9-$factorization Φ of w as shown in Figure 0.5. Statement (1) in Theorem 4.3 is now immediate, where either F_1, F_2, F_3 or, respectively, F_3, F_4, F_5 are all non-empty with respect to Φ.

   Furthermore, by Theorem 4.2, the substrings $\bar{\Phi}_2$ and $\bar{\Phi}_4$ can be "pumped"; this corresponds to "pumping" strings Φ_2, Φ_4, Φ_6, and Φ_8 thus proving statement (3) of the theorem. Statement (2) follows from the remark made above, i.e. from proposition 4.6. Note that if we substitute the set B_r for B_l in the above discussion, then a similar argument provides the other two symmetric cases in statement (1) of the theorem.

2. $1 \leq \mid S \mid \leq (4N_1 + 3)$: An easy counting argument confirms that there is at least one source node on $\bar{P}$ whose corresponding c-path passes through at least $2n_0$ $B-$nodes on $\bar{P}$. Let all the $B-$nodes associated with the above c-path be denoted by set B'; define the sets B_l' and B_r' for B' analogous to B_l and B_r respectively for $B_{\bar{P}}$. Without loss of generality, we may assume that the set B_l' is at least as large as B_r'. Clearly, $B_l' \cup B_r' = B'$, and hence B_l' contains at least n_0 nodes. The reader may note that if the labels on the directed edges out of these $B-$nodes are now marked, then an argument along the same lines as the case above (i.e. $\mid S \mid = 0$) suffices to prove the theorem (see Figure 0.4).

   Observe that the subtree rooted at this source node is also a subtree of Δ and hence contains no more than n_1 positions in w; statement (2), therefore, follows.

3. $\mid S \mid \geq 4(N_1 + 1)$: If any of the c-paths associated with the source nodes in S contains a minimum of $2n_0$ $B-$nodes from $B_{\bar{P}}$, then this case reduces to the previous one. Otherwise, there must be at least $4(N_1 + 1)$ source nodes, such that the parts of their c-paths along $\bar{P}$ contain at least one $B-$node from $B_{\bar{P}}$. If we let $\bar{B}$ to be the set of such $B-$nodes, and define $\bar{B}_l$ and $\bar{B}_r$ analogous to B_l and B_r respectively (for $B_{\bar{P}}$), then $\bar{B} = \bar{B}_l \cup \bar{B}_r$. So, without loss of generality, let $\bar{B}_l$ be the larger set. Then it is not difficult to see that there are at least $2(N_1 + 1)$ source nodes on $\bar{P}$ such that the parts of their c-paths (along $\bar{P}$) contain at

least one node in $\bar{B}_l$. Choose $2(N_1 + 1)$ such source nodes, labeled $(X_1, X_2, \ldots, X_{2(N_1+1)})$ in sequence with the property that for all $0 \leq i < 2N_1$, the source node labeled X_i is an ancestor of the one labeled X_{i+1} on path $\bar{P}$. By proposition 4.4 (with $m = 2$), there must be two pairs of nodes, labeled (X_i, X_j) and (X_k, X_l), with $1 \leq i < j < k < l \leq 2(N_1 + 1)$, such that $X_i = X_j$ and $X_k = X_l$. These pairs respectively define two *recursive subtrees* of Γ (see the shaded trees in Figure 0.5), thereby inducing a e_1−factorization of w which satisfies

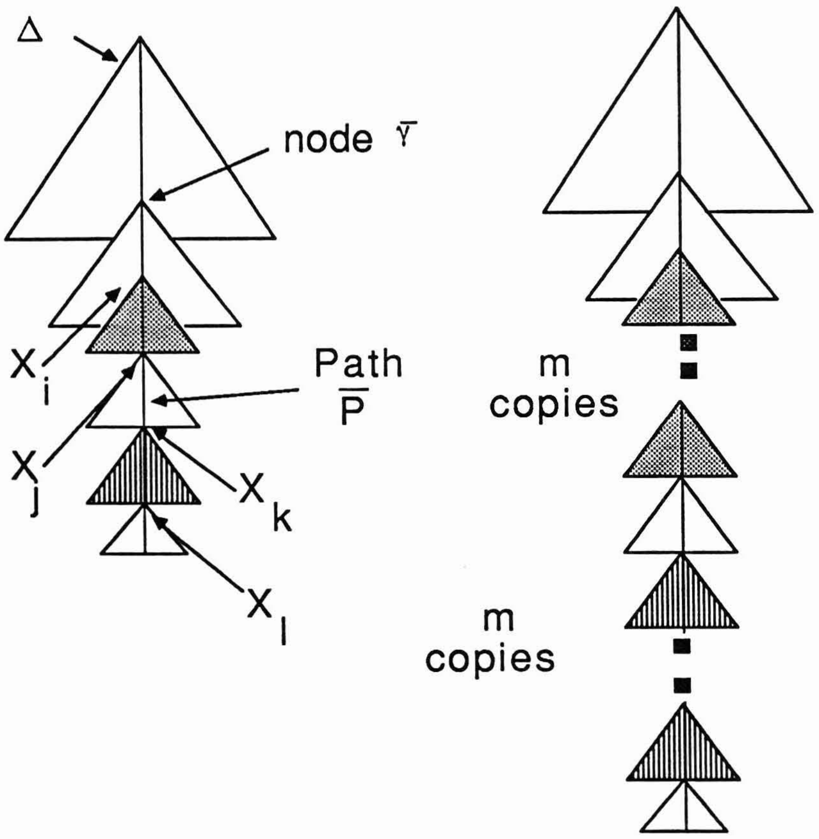

Figure 0.5: Pumping two recursive subtrees below $\bar{P}$

F_1, F_2, F_3, F_4, F_5 all non-empty (note that, in this case, we have a stronger condition than (1) in theorem 4.3). Moreover, by proposition 4.5, it is possible to "pump" both these recursive subtrees m times independently to obtain a valid derivation tree for $w^{[m]}$. Condition (2) is satisfied as before, by observing in Figure 0.5 that all "pumped" portions lie in the subtree Γ, which contains at most n_1 positions from F.

This concludes the proof of the theorem for the case $k = 1$. It can be easily extended to levels $k > 1$ in the following way. Let N_k be the number of nonterminals of G_k, and inductively let n_{k-1} be the iteration theorem constant for the control set of L generated by the sequence of grammars $G_0, G_1, \ldots, G_{k-1}$. Then the corresponding constant $n_k \equiv n(\mathcal{G})$ is given by

$n_k = {d_k}^{2n_{k-1}(2^{k+1}[N_k+1]-1)}$ where d_k is the maximum length of right hand sides of productions in G_k. The proof then follows along exactly the same lines as above, except that we use proposition 4.4 with $m = 2^k$.

5 Conclusions

Linear control grammars are promising candidates for describing natural language syntax. This paper provides another yardstick to measure the usefulness of these formalisms, viz. an estimation of the relative sizes of a context-free grammar and a linear control grammar for the same language. Our result shows that linear control grammars can be unboundedly more compact than equivalent context-free ones. In the process, we also solve another open problem by showing that the linear control language progression is a properly separated hierarchy of language classes.

Bibliography

[1] M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, MA, 1978.

[2] J. Hartmanis. On godel speed-up and succinctness of language descriptions. *Theoretical Comput. Sci.*, (26):335–342, 1983.

[3] J. Hartmanis. On the succinctness of different representations of languages. *SIAM J. Comput.*, 9(1):114–120, Feb 1980.

[4] G. T. Herman and G. Rozenberg. *Developmental Systems and Languages.* North-Holland Publishing Co., Amsterdam, 1975.

[5] O. H. Ibarra. Simple matrix languages. *Info. and Control*, 17:359–394, 1970.

[6] Hartmanis J. Context-free languages and turing machine computations. In *Proc. Symposia in Applied Math. 19*, American Mathematical Society, Providence, R.I., 1967.

[7] T. Kasai. An hierarchy between context-free and context-sensitive languages. *J. Comput. Syst. Sci.*, 4:492–508, 1970.

[8] N. A. Khabbaz. A geometric hierarchy of languages. *J. Comput. Syst. Sci.*, 8:142–157, 1974.

[9] M. A. Palis and S. Shende. Upper bounds on recognition of a hierarchy of non-context-free languages. *Theoretical Computer Science*, to appear in 1991.

[10] D. J. Rozenkrantz. Programmed grammars and classes of formal languages. *J. ACM*, 16:107–131, 1969.

[11] A. Salomaa. Matrix grammars with a leftmost restriction. *Info. and Control*, 20:143–149, 1970.

[12] J. W. Thatcher. Tree automata: An informal survey. In A. V. Aho, editor, *Currents in the Theory of Computing*, pages 143–172, Prentice Hall Inc., Englewood Cliffs, NJ, 1973.

[13] L. G. Valiant. A note on the succinctness of descriptions of deterministic languages. *Info. and Control*, 32:139–145, 1976.

[14] D. J. Weir. *Context-Free Grammars to Tree Adjoining Grammars and Beyond.* Technical Report, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1987.