JOURNAL OF COMPUTER AND SYSTEM SCIENCES 30, 86–115 (1985)

Hierarchies of Hyper-AFLs

JOOST ENGELFRIET*

Department of Computer Science, Twente University of Technology, P.O.Box 217, 7500AE Enschede, The Netherlands

Received August 1982; revised November 1, 1984

For a full semi-AFL K, $B(K)$ is defined as the family of languages generated by all K-extended basic macro grammars, while $H(K) \subseteq B(K)$ is the smallest full hyper-AFL containing K; a full basic-AFL is a full AFL K such that $B(K) = K$ (hence every full basic-AFL is a full hyper-AFL). For any full semi-AFL K, K is a full basic-AFL if and only if $B(K)$ is substitution closed if and only if $H(K)$ is a full basic-AFL. If K is not a full basic-AFL, then the smallest full basic-AFL containing K is the union of an infinite hierarchy of full hyper-AFLs. If K is a full principal basic-AFL (such as INDEX, the family of indexed languages), then the largest full AFL properly contained in K is a full basic-AFL. There is a full basic-AFL lying properly in between the smallest full basic-AFL and the largest full basic-AFL in INDEX.

INTRODUCTION

One of the main families of languages studied in formal language theory is the family of indexed languages, introduced in [1], which we will denote by INDEX. It can be defined by (at least) two families of grammars: the indexed grammars of [1] and the OI (outside–in) macro grammars of [22], both natural extensions of the context-free grammars. Moreover, it can be defined by a natural family of automata, the nested stack automata [2], which extend the pushdown automata and the one-way stack automata. Thus, since indexed languages are context-sensitive, one might claim for INDEX a position between the context-free languages and the context-sensitive languages in the Chomsky- hierarchy. Finally, from an algebraic point of view, the indexed languages can be characterized as fixed points of a set of equations in a certain algebra. From this point of view the indexed languages can be obtained from the context-free languages just as the context-free languages from the regular languages (cf. [17]). In formal language theory one wishes to know how far away INDEX is from the context-free languages in terms of operations on languages. A more general question is: is it possible to reach INDEX in a natural way from its proper subfamilies (such as the family of context-free languages or the family STACK of one-way stack languages) by operations on the

* Current address: Dept. of Mathematics and Computer Science, University of Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands.

0022-0000/85 $3.00

languages of these subfamilies? The answers to this question in the literature are all negative, and we will add some more. We now discuss some of these answers in their proper order. Since operations on languages are studied in AFL-theory [23], we freely use AFL-terminology.

INDEX is a substitution-closed full AFL [1], but in [26] it is shown that the substitution closure of STACK is properly contained in INDEX (and there is an infinite hierarchy of full AFLs between STACK and its substitution closure). In fact, it follows from the results of [26], see [28], that if K is any full semi-AFL properly contained in INDEX, then so is the substitution closure of K, i.e., INDEX cannot be reached from any of its proper sub-AFLs by the operation of substitution.

INDEX is even a (full) super-AFL, i.e., a full AFL closed under nested iterated substitution [27]. However, the smallest (full) super-AFL containing STACK is properly contained in INDEX.

Finally, INDEX is even a full hyper-AFL [11, 33], i.e., closed under iterated substitution. In [10, 11] it is shown that ETOL, one of the main families in L-system theory [32], is the smallest full hyper-AFL, and in [13] it is proved that this smallest full hyper-AFL is properly contained in INDEX. In fact, ETOL and STACK are incomparable [18] and the smallest full hyper-AFL containing STACK is properly contained in INDEX [21].

In this paper we show that iterated substitution does not help in generating INDEX, in the following strong sense:

(1) If K is a full semi-AFL properly contained in INDEX, then the smallest full hyper-AFL containing K is still properly contained in INDEX, i.e., INDEX cannot be reached from any of its proper sub-AFLs by the operation of iterated substitution. Since there is a largest full AFL properly contained in INDEX [26], it follows that this AFL is actually a full hyper-AFL.

(2) If K is a full semi-AFL properly contained in INDEX and K is not a full hyper-AFL, then there even exists an infinite hierarchy $\{K_n\}_{n \geqslant 1}$ of full hyper-AFLs such that $K \subsetneq K_1 \subsetneq K_2 \subsetneq \cdots \subsetneq \bigcup_n K_n \subsetneq$ INDEX. (Of course, (2) implies (1)).

(3) By applying (2) three times to appropriate families K, we will show the existence of four full semi-AFLs K_i $(1 \leqslant i \leqslant 4)$ such that $K_1 \subsetneq K_2 \subsetneq K_3 \subsetneq K_4 \subsetneq$ INDEX and there is an infinite hierarchy of full hyper-AFLs between K_i and K_{i+1} $(1 \leqslant i \leqslant 3)$.

Note that the existence of infinite hierarchies of full hyper-AFLs was established in [15], cf. [8]. In particular there is an infinite hierarchy of full hyper-AFLs between INDEX and the context-sensitive languages. Infinite hierarchies of space-complexity classes were shown to be (nonfull) hyper-AFLs in [37, 6].

The above results (1)–(3) are shown by studying the properties of "extended" macro grammars, introduced in [11] and generalized in [18, 7, 21]. A K-extended macro grammar (where K is a family of languages) is, roughly speaking, a (generalized) macro grammar whose nonterminals can hold languages from K rather than strings in their arguments. We consider, in particular, K-extended basic

macro grammars, where "basic" means that nonterminals are not allowed to be nested; see [21], from which we mention the following. Let $B(K)$ denote the family of languages generated by all K-extended basic macro grammars. Then, for every full semi-AFL K, $B(K)$ is a full AFL such that $K \subseteq H(K) \subseteq B(K)$, where $H(K)$ is the smallest full hyper-AFL containing K. Hence every full basic-AFL K, i.e., full AFL such that $B(K) \subseteq K$, is a full hyper-AFL (but not vice versa). It is easy to show, using macro grammars, that INDEX is a full basic-AFL.

In this paper we show that the operator B on families of languages has many of the nice properties that are abstracted in the notion of syntactic operator [28]. In particular B is "hierarchical" [30]: if a full semi-AFL K is not closed under B, i.e., it is not a full basic-AFL, then $B^n(K) \subsetneq B^{n+1}(K)$ for all $n \geqslant 0$. Hence, if K is not closed under B, then the smallest full basic-AFL containing K (denoted $B^*(K) = \bigcup_n B^n(K)$) is the union of an infinite hierarchy of full hyper-AFLs (because $B^n(K) \subseteq H(B^n(K)) \subseteq B^{n+1}(K)$). A result of this type (for substitution) was first proved by Greibach, using her well-known syntactic lemma [26]: if a full semi-AFL K is not closed under substitution, then the smallest substitution-closed full AFL containing K is the union of an infinite hierarchy of full AFLs (showing that substitution is hierarchical). From the fact that B is hierarchical, results (1) and (2) above clearly follow: in fact, if $K \subsetneq$ INDEX, then even $B^*(K) \subsetneq$ INDEX (because INDEX is a full principal basic-AFL). Proving result (3) takes more effort. We note that we prove (3) with $K_1 = \mathrm{REG}$, the family of regular languages, $K_2 = B^*(\mathrm{REG})$, the smallest full basic-AFL, and K_3, K_4 both full basic-AFLs; hence there are at least 3 different full basic-AFLs properly contained in INDEX (viz. K_2, K_3, and K_4).

It is shown in [21] that the smallest full basic-AFL $B^*(\mathrm{REG})$ is the family of languages accepted by bounded nested stack automata, i.e., nested stack automata for which there is a bound on the depth of nesting of its stacks (in fact, in [21] a general machine characterization of full basic-AFLs is given). Properness of the "basic hierarchy" $\{B^n(\mathrm{REG})\}_{n \geqslant 1}$, as shown in the present paper, implies that the bounded nested stack automata form an infinite hierarchy with respect to depth of nesting of stacks (see Proposition 6.7 of [21]; $B^n(\mathrm{REG})$ corresponds roughly to depth of nesting $n-1$).

This paper is divided into 6 sections. Section 1 contains preliminary definitions and notation. In Section 2 we recall the definitions of iteration grammar and extended basic macro grammar, and mention some of their properties. In Section 3 we consider the language of cuts, introduced in [18], which is a basic macro language not in ETOL. Some properties of this language are needed in later sections. Section 4 treats the "cut-operation": for each language L, $\mathrm{cut}(L)$ is a language obtained by mixing strings from L intimately with strings of the language of cuts. In this section the main technical result of the paper is proved: if K is a full semi-AFL closed under iterated finite substitution, then $L \in H(K) - K$ implies $\mathrm{cut}(L) \in B(K) - H(K)$. Together with results from [21] saying that, if $B(K) - H(K) \neq \varnothing$ then $H(B(K)) - B(K) \neq \varnothing$, and that $B(K)$ is closed under iterated finite substitution, this implies that the "basic hierarchy" $\{B^n(\mathrm{REG})\}_{n \geqslant 1}$ is proper. In Section 5 we prove the general result that B is hierarchical and indicate its consequences, as discussed above.

Finally, in Section 6, we prove result (3) above. The main theorem of this section says that the additional power of full basic-AFLs with respect to full hyper-AFLs does not help in copying languages. In particular, if the language $\{w \# w^R \mid w \in L\}$ is in $B^*(K)$, then it is in $H(K)$.

1. Preliminaries

We assume the reader is familiar with the basic concepts of formal language theory (e.g., [31]), in particular AFL theory [23, 9] and L-system theory [32]. In this section we fix some notation.

A *hierarchy* of sets is a family $\{A_n\}_{n \geq 1}$ of sets such that, for all n, $A_n \subseteq A_{n+1}$. The hierarchy is *infinite* if there is no $m \geq 1$ such that $A_n \subseteq A_m$ for all n; it is *proper* if $A_n \subsetneq A_{n+1}$ for all n.

The empty set is denoted $\varnothing$. For a finite set A, $\#(A)$ denotes its cardinality. The empty string is denoted λ. For any string $w = a_1 a_2 \cdots a_n$ ($n \geq 0$, a_i is a symbol), w^R denotes the reverse of w ($w^R = a_n \cdots a_2 a_1$), $|w|$ denotes the length of w ($|w| = n$), $\#_a(w)$ denotes the number of occurrences of symbol a in w ($\#_a(w) = \#(\{i \mid a_i = a\})$), and alph($w$) denotes the *alph* of w, i.e., the set of all symbols occurring in w (alph$(w) = \{a_1, a_2, ..., a_n\}$).

ONE denotes the family of singleton languages, i.e., ONE $= \{\{w\} \mid w$ is a string$\}$. FIN, REG, CF, and INDEX denote the families of finite, regular, context-free, and indexed languages [1], respectively.

Let K be a family of languages and Δ an alphabet. A *K-substitution* on Δ is a mapping $f: \Delta \to K$, extended to strings and languages in the usual way: for strings u and v, $f(uv) = f(u) \cdot f(v)$; $f(\lambda) = \{\lambda\}$; for a language L, $f(L) = \bigcup \{f(w) \mid w \in L\}$. Thus, for $L \subseteq \Delta^*$, $f(L)$ is a language, not necessarily in K. A finite substitution is a FIN-substitution. Let K_0 be a family of languages. Then Sub(K_0, K) denotes the family $\{f(L) \mid L \in K_0, f$ is a K-substitution$\}$. Note that in [23], Sub is denoted Sûb. K_0 is closed under K-substitution if Sub$(K_0, K) \subseteq K$. K is substitution closed if Sub$(K, K) \subseteq K$.

We need a particular type of substitution, called *syntactic substitution*, introduced in [25]. For languages L_1 and L_2 over disjoint alphabets Σ_1 and Σ_2, respectively, $\tau(L_1, L_2) = \{a_1 y_1 a_2 y_2 \cdots a_n y_n \mid n \geq 0, a_i \in \Sigma_1, a_1 a_2 \cdots a_n \in L_1, y_i \in L_2$ for $1 \leq i \leq n\}$. For arbitrary L_1 and L_2, we assume that $\tau(L_1, L_2)$ involves an implicit change of alphabets such that the alphabets of L_1 and L_2 are disjoint (this should not lead to problems: L_1 and L_2 will always be taken from families of languages which allow such a change of alphabet).

An *ngsm mapping* is a mapping realized by a nondeterministic generalized sequential machine with accepting states (ngsm). Let K be a family of languages. K is a *full semi-AFL* if it contains REG and is closed under ngsm mappings, inverse ngsm mappings, and union. $\hat{S}(K)$ denotes the smallest full semi-AFL containing K. A full semi-AFL K is full principal if, for some $L \in K$, $\hat{S}(L) = K$ (where $\hat{S}(L)$ stands for $\hat{S}(\{L\})$); L is called a generator of K. A full AFL is a full semi-AFL closed

under concatenation and Kleene star. $\hat{F}(K)$ denotes the smallest full AFL containing K, and $\hat{F}_\sigma(K)$ denotes the smallest substitution-closed full AFL containing K. If K_0 and K are full semi-AFLs, then so is $\mathrm{Sub}(K_0, K)$ [23].

Finally we need the concept of a macro grammar. For more formal definitions see [22 or 17]. A ranked alphabet Δ is a finite set of symbols such that with each symbol $A \in \Delta$ a unique nonnegative integer (the rank of A) is associated. For $i \geqslant 0$, Δ_i denotes the set of all symbols of rank i in Δ. A *macro grammar* $G = (F, \Sigma, X, S, P)$ consists of a ranked alphabet F of nonterminals, a terminal alphabet Σ, a finite set $X = \{x_1, ..., x_m\}$ of variables, where m is the maximal rank of a symbol in F (F, Σ, and X are mutually disjoint), an initial nonterminal $S \in F_0$, and a finite set of productions or rules of the form $A(x_1, ..., x_n) \to t$, where $A \in F_n$ and t is a term over $F \cup \Sigma \cup \{x_1, ..., x_n\}$ (each element of $F_0 \cup \Sigma \cup \{x_1, ..., x_n\}$ is a term; if t_1 and t_2 are terms, then $t_1 t_2$ is a term; if $B \in F_k$ and $t_1, ..., t_k$ are terms, then $B(t_1, ..., t_k)$ is a term). We will always use a macro grammar in the outside–in (OI) mode of derivation, i.e., the above production $A(x_1, ..., x_n) \to t$ can be applied only to an occurrence of A that is not nested in another nonterminal. Application of this production consists of replacing a subterm of the form $A(t_1, ..., t_n)$, where t_i is a term, by the result of substituting t_i for x_i in t for all i, $1 \leqslant i \leqslant n$. Productions are applicable to terms over $F \cup \Sigma$, but, if needed, also to terms over $F \cup \Sigma \cup X$. The language generated by G is $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$ as usual. The family of languages generated by all macro grammars is INDEX [22].

2. Iteration Grammars and Extended Basic Macro Grammars

In this section we recall the definitions of iteration grammar, full hyper-AFL, extended basic macro grammar, and full basic-AFL. We also state some of their properties. We start with iteration grammars, see [32].

Let K_0 and K be families of languages. A (K_0, K)-*iteration grammar* is a construct $G = (V, \Sigma, U, A)$, where V is an alphabet, $\Sigma \subseteq V$ is the terminal alphabet, U is a finite set of K-substitutions on V (such that $f(a) \subseteq V^*$ for every $f \in U$ and $a \in V$), and $A \in K_0$ is a language over V (the set of axioms). The language generated by G is $L(G) = U^*(A) \cap \Sigma^*$, where $U^*(A) = \bigcup \{f_n(...f_2(f_1(A))...) \mid n \geqslant 0, f_i \in U\}$.

Derivations of G are defined as usual. Let v, w, w_i be strings over V. If $w \in f(v)$ for some $f \in U$, then we write $v \Rightarrow w$, or $v \overset{f}{\Rightarrow} w$ to indicate by which substitution the derivation step is made. A derivation is a sequence $w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$ ($n \geqslant 1$) such that there exists $f_1, f_2, ..., f_{n-1} \in U$ with $w_{i+1} \in f(w_i)$; we also write $w_1 \overset{*}{\Rightarrow} w_n$ or $w_1 \overset{\pi}{\Rightarrow} w_n$, where $\pi = f_1 f_2 \cdots f_{n-1} \in U^*$, and we say that the derivation $w_1 \overset{*}{\Rightarrow} w_n$ has *control string* π (for $n = 1$, $w_1 \overset{*}{\Rightarrow} w_1$ has control string λ). Thus $L(G) = \{w \in \Sigma^* \mid v \overset{*}{\Rightarrow} w \text{ for some } v \in A\}$. If $M \subseteq U^*$, then $L(G, M) = \{w \in \Sigma^* \mid v \overset{\pi}{\Rightarrow} w \text{ for some } v \in A \text{ and } \pi \in M\}$; note that $L(G, M) = M(A) \cap \Sigma^*$, where $M(A)$ is defined similarly to $U^*(A)$. M is called a *control language* for G. A *sentential form* of G is a string $w \in V^*$ such that $v \overset{*}{\Rightarrow} w$ for some $v \in A$.

A *K-iteration grammar* is a (K, K)-iteration grammar. Note that if K contains

$\{S\}$ for every symbol S, then we may always assume, for a K-iteration grammar $G=(V, \Sigma, U, A)$, that $A=\{S\}$ for some $S \in V$ (add a new substitution f to U such that $f(S)=A$). A FIN-iteration grammar is also called an *ETOL system.* A K-iteration scheme is a triple $G=(V, \Sigma, U)$, just as in a K-iteration grammar but without a set of axioms; in particular, a FIN-iteration scheme is called an *ETOL scheme.*

The family of all languages generated by (K_0, K)-iteration grammars is denoted $H(K_0, K)$. This should not be confused with the notation in [7], where K_0 is the family of control languages. We denote $H(K, K)$ also by $H(K)$, $H(K, \text{FIN})$ by $E(K)$, $H(\text{FIN})=E(\text{FIN})$ by ETOL, and $H(\text{ONE})$ by EDTOL.

We say that K_0 is *closed under iterated K-substitution* if $H(K_0, K) \subseteq K_0$; in particular, K_0 is closed under iterated finite substitution if $E(K_0) \subseteq K_0$. K is closed under iterated substitution if $H(K) \subseteq K$. K is a *full hyper-AFL* if it is a full AFL closed under iterated substitution (i.e., $H(K) \subseteq K$). Note that every full hyper-AFL is substitution closed. $\hat{H}(K)$ denotes the smallest full hyper-AFL containing K. We will need the following facts.

2.1. THEOREM. *Let K be a full semi-*AFL.

(1) $H(K)=\hat{H}(K)$

(2) $E(K)$ *is a full semi-*AFL

(3) *If $G=(V, \Sigma, U, A)$ is a (K, FIN)-iteration grammar (thus, $L(G) \in E(K)$) and M is a regular control language, then $L(G, M) \in E(K)$.*

Proof. (1) is Theorem 6.3 of [4]. The proofs of (2) and (3) are left to the reader. They can be obtained by generalizing the proofs of Lemma 4.3 and Theorem 2.1 of [4], respectively. ▮

A *full super-AFL* (introduced in [27], without the adjective "full") is a full AFL closed under iterated nested substitution (a substitution f is nested if $a \in f(a)$ for every symbol a). We use $A(K)$ to denote the smallest full super-AFL containing K. Actually, $A(K)$ can be defined as the class of languages generated by "K-extended context-free grammars" and then shown to be the smallest full super-AFL containing K, by a proof similar to that of Theorem 2.1(1), see [36, 38, 5]. Note that every full hyper-AFL is a full super-AFL, and every full super-AFL is substitution closed.

We now turn to extended basic macro grammars (see [21, 7]). Let K be a family of languages. A *K-extended basic macro grammar* is a construct $G=(F, \Psi, \Sigma, X, S, d, P)$, where F is a ranked alphabet of nonterminals, Ψ is a ranked alphabet of *language names*, Σ is the terminal alphabet, $X=\{x_1, \ldots, x_m\}$ is a finite set of variables, where m is the maximal rank of a symbol in $F \cup \Psi$ (F, Ψ, Σ, and X are mutually disjoint), $S \in F_0$ is the initial nonterminal, d is a mapping $\Psi \to K$ such that, for $\psi \in \Psi_n$, $d(\psi) \subseteq (\Sigma \cup \{x_1, \ldots, x_n\})^*$, $d(\psi)$ is the *domain* of ψ, and P is a finite set of productions or rules each of the form

$$A(\mathbf{x}) \to \psi_1(\mathbf{x})\, B_1(\phi_1(\mathbf{x}))\, \psi_2(\mathbf{x})\, B_2(\phi_2(\mathbf{x})) \cdots B_k(\phi_k(\mathbf{x}))\, \psi_{k+1}(\mathbf{x}),$$

where $(\mathbf{x}) = (x_1, ..., x_n)$, $n \geq 0$, $A \in F_n$, $k \geq 0$, $B_i \in F$, $\psi_i \in \Psi_n$, and $(\phi_i(\mathbf{x})) = (\psi_{i1}(\mathbf{x}), \psi_{i2}(\mathbf{x}), ..., \psi_{is}(\mathbf{x}))$ with $\psi_{ij} \in \Psi_n$ and s is the rank of B_i. G is *linear* if $k = 0$ or $k = 1$ in each production. Whenever $d(\psi)$ is a singleton $\{w\}$, we use w rather than $\psi(\mathbf{x})$, i.e., elements of ONE are displayed without using language names.

With each K-extended basic macro grammar $G = (F, \Psi, \Sigma, X, S, d, P)$ we associate an ordinary macro grammar G' with an *infinite* number of rules, by viewing the language names as nonterminals, as follows:

$$G' = (F \cup \Psi, \Sigma, X, S, P'),$$

where

$$P' = P \cup \{\psi(x_1, ..., x_n) \to w \mid n \geq 0,\ \psi \in \Psi_n,\ w \in d(\psi)\}.$$

By definition, the derivations of G are those of G'. A sentential form is a term t such that $S \overset{*}{\Rightarrow} t$. The language generated by G is defined by $L(G) = L(G')$. Note that G, without d, may be viewed as an operation on languages: $L(G)$ is the result of applying this operation to the languages $d(\psi)$, $\psi \in \Psi$.

The family of all languages generated by K-extended {linear} basic macro grammars is denoted $B(K)$ {$LB(K)$, respectively}. Note that $B(\text{ONE})$ is the family of basic macro languages [22] and $B(\text{FIN}) = B(\text{REG})$ is the family EB of "extended basic macro languages" (accepted by stack-pushdown machines) of [18]. A *full basic-AFL* is a full AFL K such that $B(K) \subseteq K$. INDEX is a full (principal) basic-AFL (Corollary 2.5 of [21]). $\hat{B}(K)$ denotes the smallest full basic-AFL containing K. $B^*(K)$ denotes $\bigcup \{B^n(K) \mid n \geq 1\}$.

In the next theorem we collect a few useful facts (for the syntactic substitution τ, see Section 1).

2.2. THEOREM. *Let K be a full semi-*AFL:

(1) $K \subseteq E(K) \subseteq H(K) = LB(K) \subseteq B(K)$

(2) $B(K)$ *is a full* AFL *closed under iterated $LB(K)$-substitution*

(3) $B^*(K) = \hat{B}(K)$, *in particular* $B^*(\text{FIN}) = B^*(\text{REG})$ *is the smallest full basic-*AFL

(4) *For any two languages L_1 and L_2, if $\tau(L_1, L_2) \in B(K)$, then $L_1 \in CF$ or $L_2 \in LB(K)$.*

Proof. The inclusions in (1) are obvious. $H(K) = LB(K)$ is shown in Theorem 3.5 of [7]. For (2), see Theorems 3.4 and 3.6 of [21]; (3) follows from (2), cf. Section VI.3 of [9], and (4) is Theorem 3.7 of [21] (note that $\tau(L_1, L_2)$ involves a change of alphabet; since both CF and $LB(K)$ are closed under change of alphabet, L_1 or L_2 can be reobtained). ∎

Note that, by Theorem 2.2(1), every full basic-AFL is a full hyper-AFL.

3. The Language of Cuts

In this section we consider the language of cuts, introduced in [18], which will be of vital importance in the next sections. A *cut* is a sequence of nodes which form a cross section of the infinite binary tree (see Fig. 1). Each node is coded in Dewey notation by the path by which it can be reached from the root (the cut suggested in Fig. 1 is $\langle 00, 01, 100, 101, 11\rangle$). Formally, a cut is a finite nonempty sequence of strings over $\{0, 1\}$ defined recursively as follows:

(i) $\langle\lambda\rangle$ is a cut

(ii) if $\langle w_1,..., w_i,..., w_n\rangle$ is a cut$(1 \leqslant i \leqslant n)$, then $\langle w_1,..., w_i0, w_i1,..., w_n\rangle$ is a cut.

The strings w_i in a cut $\langle w_1,..., w_n\rangle$ are called *nodes*. The following properties of cuts are obvious.

(C1). All nodes in a cut are different.

(C2). A proper subsequence of a cut is not a cut, i.e., if $\langle w_1,..., w_n\rangle$ is a cut and $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n$ with $k < n$, then $\langle w_{i_1},..., w_{i_k}\rangle$ is not a cut.

(C3). For every $n \geqslant 1$ there are finitely many cuts with n nodes.

(C4). For every $k \geqslant 1$ there is a cut $\langle w_1,..., w_n\rangle$ with $n \geqslant 3$ such that $|w_i| \geqslant k$ for all i, $1 \leqslant i \leqslant n$.

We now define the corresponding language (over $\{\$, 0, 1\}$).

3.1. Definition. The *language of cuts*, denoted CUT, is

$$\{\$w_1\$w_2 \cdots \$w_n \mid \langle w_1,..., w_n\rangle \text{ is a cut}\}.$$

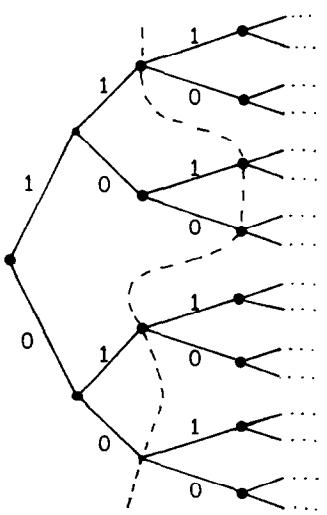

Fig. 1. An example of a cut of the infinite binary tree.

Note that CUT $\in B$(ONE), i.e., CUT is an ordinary basic macro language. In fact, it is generated by the basic macro grammar with productions $S \to A(\$)$, $A(x) \to A(x0)\, A(x1)$, $A(x) \to x$. It is shown in [18] that a particular sublanguage of CUT is not in ETOL, and is not even a tree transformation language. Here, we want to show the same for the language CUT itself. Let yT(REC) denote the family of tree transformation languages (i.e., the yields of images of recognizable tree languages under nondeterministic top-down tree transducers, see, e.g., [16]). The next theorem is (for ETOL) a slightly simplified version of Theorem V.2.1 of [32]. We repeat, however, (our version of) the proof, because a similar argument will be used in the proof of Theorem 3.4.

3.2. THEOREM. *Let L be a language over alphabet Σ with the following two properties (where $\$ \in \Sigma$).*

(i) *For every $n \geqslant 1$ the language $\{w \in L \mid \#_{\$}(w) = n\}$ is finite.*

(ii) *For every $k \geqslant 1$ there is a string $w_0 \$ w_1 \$ w_2 \cdots \$ w_n \in L$ with $n \geqslant 2$, $w_i \in (\Sigma - \{\$\})^*$, and $|w_i| \geqslant k$ for all i, $1 \leqslant i \leqslant n-1$.*

Then $L \notin$ ETOL and even $L \notin yT$(REC).

Proof. We first prove that $L \notin$ ETOL and then indicate how to extend this proof to yT(REC).

Let $G = (V, \Sigma, U, \{S\})$ be an ETOL system with $S \in V$ such that $L(G) \subseteq L$. We will show that $L(G) \subsetneq L$. As in the proof of Theorem V.2.1 of [32] we first change G in such a way that to every symbol b information is added indicating whether it will generate no \$, exactly one occurrence of \$, or at least two occurrences of \$ (coded as $\langle b, 0\rangle$, $\langle b, 1\rangle$, and $\langle b, 2\rangle$, respectively). The construction is standard: $G_1 = (V_1, \Sigma, U_1, A_1)$ with $V_1 = (V \times \{0, 1, 2\}) \cup \Sigma$ and $A_1 = \{\langle S, 0\rangle, \langle S, 1\rangle, \langle S, 2\rangle\}$. For each $f \in U$, U_1 contains a finite substitution f_1 such that, for $b \in V$ and $i \in \{0, 1, 2\}$, $f_1(\langle b, i\rangle) = \{\langle a_1, i_1\rangle\langle a_2, i_2\rangle \cdots \langle a_n, i_n\rangle \mid a_j \in V,\ a_1 a_2 \cdots a_n \in f(b)$ and $i = i_1 + \cdots + i_n$ if $i_1 + \cdots + i_n \leqslant 1$, $i = 2$ otherwise$\}$, and, for $b \in \Sigma$, $f_1(b) = \varnothing$. Finally, U_1 contains an additional finite substitution f such that $f(\langle \$, 1\rangle) = \{\$\}$, $f(\langle b, 0\rangle) = \{b\}$ for all $b \in \Sigma - \{\$\}$, and $f(a) = \varnothing$ for all other $a \in V_1$. Clearly $L(G_1) = L(G)$ and, for every $b \in V$, $\langle b, 0\rangle$ generates no \$, $\langle b, 1\rangle$ generates exactly one \$, and $\langle b, 2\rangle$ generates at least two \$s.

Consider a sentential form w of G_1 such that alph$(w) \subseteq V \times \{0, 1\}$, i.e., w consists of symbols generating at most one \$. It follows from property (i) of L that w generates finitely many strings over Σ. We now show that we may assume a bound on the length of derivations of terminal strings from any sentential form with the same alph as w. Let alph$(w) = \{a_1, a_2, \ldots, a_r\}$, and consider the set W of r-tuples of languages over Σ defined by $W = \{\langle W_1, \ldots, W_r\rangle \mid$ there exists $\pi \in U_1^*$ such that, for every i $(1 \leqslant i \leqslant r)$, $W_i = \{u \in \Sigma^* \mid a_i \Rightarrow^{\pi} u\}$ and $W_i \neq \varnothing\}$. Intuitively $\langle W_1, \ldots, W_r\rangle$ indicates which strings can be derived from $a_1, \ldots, a_r$ for a given control string π. Since w generates finitely many strings over Σ, W is a finite set: if $u \in W_i$, then u is a substring of a string generated by w. Hence we may restrict the π in the definition of

W to be of length $\leqslant q$ for some fixed integer q. Now consider any other string v with $\text{alph}(v) = \text{alph}(w)$, and let $v \Rightarrow^{\pi} u \in \Sigma^*$ for some $\pi \in U_1^*$. It follows that there exists $\pi' \in U_1^*$ with $|\pi'| \leqslant q$ and $v \Rightarrow^{\pi'} u$. Hence we may put a bound q on the length of all derivations from a string with the same alph as w. Since this argument holds for any w with $\text{alph}(w) \subseteq V \times \{0, 1\}$, we conclude that there is a uniform upper bound, say p, on the length of derivations we have to consider, starting from sentential forms over $V \times \{0, 1\}$.

Let m be the maximal length of strings in $f(b)$, $f \in U_1$, and $b \in V_1$. Consider, in a derivation of a string $u \in L$ (with at least two occurrences of \$), the last sentential form v which is not over $V \times \{0, 1\}$. By the above, u can be derived from v in at most $p+1$ steps. Consider a symbol $\langle b, 2\rangle$ in v. In the next $p+1$ (or less) steps it derives at most m^{p+1} symbols. Hence v contains a substring $\$w\$$ with $|w| \leqslant m^{p+1}$. Consequently every string (with at least two \$) generated by G_1 contains a substring $\$w\$$ with $|w| \leqslant m^{p+1}$. However, by property (ii) there is a string $w_0\$w_1\$w_2 \cdots \$w_n$ in L with $n \geqslant 2$, $w_i \in (\Sigma - \{\$\})^*$, and $|w_i| > m^{p+1}$ for all i, $1 \leqslant i \leqslant n-1$. Hence this string of L cannot be generated by G_1, i.e., $L(G_1) \subsetneq L$. Thus $L \notin$ ETOL.

The proof that $L \notin yT(\text{REC})$ is left to the reader. It consists of an obvious generalization of the above argument to top-down tree transducers (for the way in which top-down tree transducers generalize ETOL systems, see, e.g., [16]). We just note that (in the terminology of [16]) we have to consider the state-set of a node of the input tree instead of $\text{alph}(w)$. We then find a uniform upper bound on the size of the input subtrees we have to consider for nodes with state-set in $Q \times \{0, 1\}$, where Q is the set of states of the top-down tree transducer under consideration. ∎

3.3. COROLLARY. CUT $\notin$ ETOL *and even* CUT $\notin yT$(REC).

Proof. Properties (C3) and (C4) of the set of all cuts imply properties (i) and (ii) of CUT (in Theorem 3.2). ∎

In the next theorem we generalize the fact that CUT $\notin$ ETOL: the operation of iterated finite substitution does not help in generating CUT. This is also true for other languages with properties stronger than those of Theorem 3.2. For a language L over Σ (with $\$ \in \Sigma$) and $k \geqslant 1$, let L_k denote the language $\{w_0\$w_1\$w_2 \cdots \$w_n \in L \mid w_i \in (\Sigma - \{\$\})^*$ and $|w_i| \geqslant k$ for all i, $1 \leqslant i \leqslant n-1\}$. Property (ii) of Theorem 3.2 expresses that L_k contains a string (with $n \geqslant 2$) for all k. In the next theorem we keep property (i), but strengthen property (ii) of Theorem 3.2.

3.4. THEOREM. *Let K be a full semi-*AFL. *Let L be a language over Σ with the following two properties (where $\$ \in \Sigma$).*

(i) *For every $n \geqslant 1$ the language $\{w \in L \mid \#_{\$}(w) = n\}$ is finite.*

(ii) *For every $k \geqslant 1$, $L \in \hat{S}(L_k)$.*

If $L \in E(K)$, then $L \in K$.

Proof. The proof first follows the proof of Theorem 3.2. Let $G=(V, \Sigma, U, A)$ be a (K, FIN)-iteration grammar such that $L(G)=L$. Change G into $G_1=(V_1, \Sigma, U_1, A_1)$ with $L(G_1)=L(G)$, as in the proof of Theorem 3.2. The only difference is that $A_1=\{\langle a_1, i_1\rangle\langle a_2, i_2\rangle \cdots \langle a_n, i_n\rangle \in (V\times\{0,1,2\})^* \mid a_1 a_2 \cdots a_n \in A\}$. Clearly $A_1 \in K$. As in the proof of Theorem 3.2 (using property (i)) we find a uniform upper bound p on the necessary length of derivations starting from sentential forms over $V\times\{0,1\}$. If a string $v \in A_1$ contains a symbol $\langle b, 2\rangle$, $b \in V$, then every terminal string generated by v contains a substring $\$w\$$ with $|w| \leqslant m^{p+1}$. Let $k=1+m^{p+1}$. Hence all strings of L_k are generated from strings in $A_1 \cap (V\times\{0,1\})^*$ and, in fact, by derivations of length $\leqslant p$. Therefore, L_k can be obtained from A_1 by first intersecting with the regular language $(V\times\{0,1\})^*$, then applying finite substitutions (all sequences of FIN-substitutions in U_1^* of length $\leqslant p$), taking the union of the resulting languages, intersecting with Σ^*, and finally intersecting with the regular language $\{w_0\$w_1 \cdots \$w_n \mid w_i \in (\Sigma-\{\$\})^*$ and $|w_i| \geqslant k$ for all i, $1 \leqslant i \leqslant n-1\}$. Hence $L_k \in K$. Since, by property (ii), L can be obtained from L_k by full semi-AFL operations, $L \in K$. ∎

3.5. COROLLARY. *Let K be a full semi-*AFL. *If* $\mathrm{CUT} \in E(K)$, *then* $\mathrm{CUT} \in K$.

Proof. It suffices to show that, for every $k \geqslant 1$, CUT is in the smallest full semi-AFL containing CUT_k. In fact, CUT can be obtained from CUT_k by the (deterministic) ngsm which erases from each string in CUT_k all nodes that do not start with 0^k, and erases 0^k from all nodes that do start with 0^k. Note that all cuts through the infinite binary subtree with root 0^k are initial part of some cut in CUT_k. ∎

Another example of a language L satisfying the assumptions of Theorem 3.4 is $L=\{(\$a^n)^m \mid m \geqslant n \geqslant 1\}$, cf. Corollary V.2.2 of [32].

4. THE CUT OPERATION ON LANGUAGES

The language CUT is typical for what can be done by B but not by H. The aim of this section is to show that for certain full semi-AFLs K that are not full basic-AFLs, there is a language in $B(K)$ which is not in $H(K)$. To do this we mix the strings of a language L in $B(K)-K$ with the strings of CUT. The resulting language is called $\mathrm{cut}(L)$. It will be shown that $\mathrm{cut}(L)$ is in $B(K)-H(K)$ for every full semi-AFL K closed under iterated finite substitution (i.e., $E(K) \subseteq K$). We now define $\mathrm{cut}(L)$.

4.1. DEFINITION. Let L be a language over an alphabet Δ. Let $\bar{\Delta}=\{\bar{a} \mid a \in \Delta\}$ and let \$, $\bar{\$}$, 0, 1, 2, 3 be symbols not in Δ. Let t be the homomorphism with $t(0)=2$ and $t(1)=3$. Let, for $y=a_1 a_2 \cdots a_k$ $(a_i \in \Delta)$ and $w \in \{2,3\}^*$, $y[w]$ denote the string $a_1 w\bar{a}_1 a_2 w\bar{a}_2 \cdots a_k w\bar{a}_k$. Let, finally, $\Sigma=\Delta \cup \bar{\Delta} \cup \{\$, \bar{\$}, 0, 1, 2, 3\}$. Then $\mathrm{cut}(L)=$

$\{\$w_1\bar{\$}y_1[t(w_1)]\ \$w_2\bar{\$}y_2[t(w_2)]\cdots\$w_n\bar{\$}y_n[t(w_n)]\in\Sigma^* \mid n\geqslant 1,\ \langle w_1, w_2,..., w_n\rangle$ is a cut, and $y_1, y_2,..., y_n\in L\}$.

The notation of Definition 4.1 will be used throughout this section. With respect to cuts, we use the terminology of Section 3; in particular, $w_1,..., w_n$ are the nodes of the cut$\langle w_1,..., w_n\rangle$ corresponding to the string $\$w_1\bar{\$}y_1[t(w_1)]\cdots\$w_n\bar{\$}y_n[t(w_n)]\in L$.

Note that $\text{cut}(L)\subseteq(\$\{0, 1\}^*\bar{\$}(\Delta\{2, 3\}^*\bar{\Delta})^*)^*$. If, as an example, *abb* and *ba* are in L, then $\$00\bar{\$}a22\bar{a}b22\bar{b}b22\bar{b}\$01\bar{\$}b23\bar{b}a23\bar{a}\$1\bar{\$}b3\bar{b}a3\bar{a}$ is in $\text{cut}(L)$.

We first show that $B(K)$ is closed under the cut operation.

4.2. LEMMA. *Let K be a family of languages containing* ONE *and closed under homomorphisms, and let L be a language over Δ. If $L\in B(K)$, then* $\text{cut}(L)\in B(K)$.

Proof. Clearly, $B(K)$ contains ONE and is closed under homomorphisms. Let $L\in B(K)$. We first exhibit a $B(K)$-extended basic macro grammar G generating $\text{cut}(L)$. Let $G=(F, \Psi, \Sigma, X, S, d, P)$ with $F=\{S, A, D\}$, $\Psi=\{\psi\}$, A has rank 2, D and ψ have rank 1, $X=\{x_1, x_2\}$, $d(\psi)=h(L)$, where h is the homomorphism such that $h(a)=ax_1\bar{a}$ for every $a\in\Delta$, and P contains the following productions (recall that singleton languages are just displayed, without using language names):

$$S\to A(\lambda, \lambda)$$
$$A(x_1, x_2)\to A(x_1 0, x_2 2)\, A(x_1 1, x_2 3)$$
$$A(x_1, x_2)\to\$x_1\bar{\$}D(x_2)$$
$$D(x_1)\to\psi(x_1).$$

To see that $L(G)=\text{cut}(L)$, note that the sentential forms of G involving A's only are of the form $A(w_1, t(w_1))\, A(w_2, t(w_2))\cdots A(w_n, t(w_n))$, where $\langle w_1,..., w_n\rangle$ is an arbitrary cut. By the last two productions, $A(w, t(w))$ produces any string $\$w\bar{\$}y[t(w)]$ with $y\in L$.

Since $h(L)\in B(K)$, there is a K-extended basic macro grammar G_ψ generating $h(L)$, with terminal alphabet $\Delta\cup\bar{\Delta}\cup\{x_1\}$. Let S_ψ be the initial nonterminal of G_ψ. By Lemma 2.4 of [21] we can turn the terminal symbol x_1 into a variable, i.e., there is a K-extended basic macro grammar G'_ψ with terminal alphabet $\Delta\cup\bar{\Delta}$ such that $h(L)=\{w\in(\Delta\cup\bar{\Delta}\cup\{x_1\})^* \mid S_\psi(x_1)\overset{*}{\Rightarrow} w \text{ in } G'_\psi\}$. Let G' be the K-extended basic macro grammar with all productions of both G and G'_ψ, except that production $D(x_1)\to\psi(x_1)$ is changed into $D(x_1)\to S_\psi(x_1)$. It should be clear that $L(G')=L(G)$, and hence $\text{cut}(L)\in B(K)$. ∎

Note that in particular $B(\text{ONE})$, the family of (ordinary) basic macro languages, is closed under the cut operation. We need two elementary properties of the language $\text{cut}(L)$, expressed in the next lemma (in which we use the notation of Definition 4.1).

4.3. LEMMA. *Let L be a language over alphabet Δ such that, for every string $w \in L$, $|w| \geq 2$ and $w \notin \Delta^* aa \Delta^*$ for any $a \in \Delta$. Let $N = \{u_1 a w \bar{a} u_2 \in \Sigma^* \mid u_1, u_2 \in \Sigma^*$ and either $a = \$$ and $w \in \{0, 1\}^*$, or $a \in \Delta$ and $w \in \{2, 3\}^*\}$. Then* cut(L) *has the following two properties.*

4.3(i) *For all $u, v, x, y, z \in \Sigma^*$, if $xuyvz$, $xuyuz$, and $xvyvz$ are in* cut(L) *and u or v is in N, then $uyv \in (\Delta \cup \bar{\Delta} \cup \{2, 3\})^*$.*

4.3(ii). *For all $u, v, x, y \in \Sigma^*$, if xuy and xvy are in* cut(L), *$u \notin N$, and* alph$(v) \subseteq$ alph(u), *then $u = v$.*

Proof. (i) Suppose that $u \in N$ (the proof for $v \in N$ is symmetric), i.e., $u = u_1 a w \bar{a} u_2$. Clearly $a \neq \$$, because otherwise the cut corresponding to $xuyuz \in L$ would contain the same node twice (see property (C1) in Sect. 3). Hence $a \in \Delta$ and $w \in \{2, 3\}^*$. Consider $xuyuz = x u_1 a w \bar{a} u_2 y u_1 a w \bar{a} u_2 z$ and its substring $u_2 y u_1$. If $u_2 y u_1$ contains an occurrence of a symbol in $\{\$, \bar{\$}, 0, 1\}$, then the cut corresponding to $xuyuz$ contains the same node twice, contradicting (C1). Hence $u_2 y u_1 \in (\Delta \cup \bar{\Delta} \cup \{2, 3\})^*$ and so $uy = u_1 a w \bar{a} u_2 y \in (\Delta \cup \bar{\Delta} \cup \{2, 3\})^*$. It remains to show that $v \in (\Delta \cup \bar{\Delta} \cup \{2, 3\})^*$. If not, then the cut corresponding to $xuyuz$ would be a proper subsequence of the cut corresponding to $xuyvz$, contradicting property (C2) of Section 3. To see this, note that, since v can be replaced by u in $xuyvz$ without leaving cut(L), and $u \in (\Delta \cup \bar{\Delta} \cup \{2, 3\})^*$, v has to contain at least one \$ and one $\bar{\$}$, and hence the cut of $xuyuz$ contains fewer nodes than the one of $xuyvz$ (the nodes of v are removed).

(ii) Since $u \notin N$, u is of one of the forms $w_1 \bar{a}_1 a_2 w_2$, $w_1 \bar{a}_1$, $a_2 w_2$, or w_1, with $w_1, w_2 \in \{0, 1, 2, 3\}^*$, $a_1, a_2 \in \Delta \cup \{\$\}$, and by the conditions on L, $a_1 \neq a_2$ in $w_1 \bar{a}_1 a_2 w_2$. Since alph$(v) \subseteq$ alph(u), $v \notin N$ (if $v \in N$, then alph(v) contains a and $\bar{a}$ for some $a \in \Delta \cup \{\$\}$). Hence v is also of one of the above four forms. Moreover, comparison of xuy and xvy shows that u and v have to be of the same form, and that the a_i have to be the same in u and v, if they are present ($\bar{a}_1$ is determined by a_1, and a_2 by $\bar{a}_2$). Since each string of L has length ≥ 2, each node w of the cut corresponding to xuy occurs at least three times in xuy (as w, or as $t(w)$). Since changing xuy into xvy can influence at most two such occurrences of a node, u has to be equal to v. ∎

We are now ready to prove our main technical result.

4.4. THEOREM. *Let K be a full semi-*AFL, *and let L be a language. If* cut$(L) \in H(K)$, *then $L \in E(K)$.*

Proof. Let $L \subseteq \Delta^*$ and let $G = (V, \Sigma, U, \{S\})$ be a K-iteration grammar generating cut(L), where $\Sigma = \Delta \cup \bar{\Delta} \cup \{\$, \bar{\$}, 0, 1, 2, 3\}$ and $S \in V$. Since both $H(K)$ and $E(K)$ are full semi-AFLs (by Theorem 2.1), we may assume that L satisfies the conditions of Lemma 4.3, and, consequently, that cut(L) has properties 4.3(i) and (ii).

The formal proof that $L \in E(K)$ will be divided into two parts, one showing the

truth of the following statement (P) and the other using statement (P). Let k be the fixed integer $k = (\#(V)+1)^2 + \#(V)$. The reason for choosing this value of k will become clear later.

Statement (P). For each $y \in L$ there exist a cut$\langle w_1,..., w_n\rangle$ and a derivation in G of a string $v \in \text{cut}(L)$ of the form $v = \$w_1\$y[t(w_1)]\$w_2\$y[t(w_2)]\cdots\$w_n\$y[t(w_n)]$ such that this derivation is of the form

$$S \overset{*}{\Rightarrow} u_1 a u_2 \Rightarrow u_1' b_1 \cdots b_m u_2' \overset{*}{\Rightarrow} u_1'' v_1 \cdots v_m u_2'' = v \qquad (\text{P}^*)$$

(with the u's and v's in V^*, the a and b's in V, and $u_1 \Rightarrow u_1' \overset{*}{\Rightarrow} u_1''$, $u_2 \Rightarrow u_2' \overset{*}{\Rightarrow} u_2''$, $a \Rightarrow b_1 \cdots b_m$, and $b_i \overset{*}{\Rightarrow} v_i$ for $1 \leqslant i \leqslant m$) and $\#(\{i \mid \$ \in \text{alph}(v_i) \text{ and } 1 \leqslant i \leqslant m\}) > k$.

In other words statement (P) says that for some (large) k we can find for every $y \in L$ a derivation of some $\$w_1\$y[t(w_1)]\cdots\$w_n\$y[t(w_n)]$ such that in some step of that derivation some symbol produces more than k symbols which generate each at least one \$.

In the first part of the proof we will show that (P) is true by constructing an ETOL system for CUT, under the assumption that (P) is false (see Corollary 3.3). In fact, if (P) is false, then the number of \$-generating symbols produced by one symbol in one derivation step is bounded by k (in all derivations involving one y only). This will provide an upper bound for the length of strings in the substitutions of an iteration grammar generating the language of cuts.

In the second part of the formal proof we will show, roughly stated, how, using properties 4.3(i) and (ii) of cut(L), an ETOL scheme (V, Σ, U') and a language $A \in K$ can be constructed such that the corresponding (K, FIN)-iteration grammar $G' = (V, \Sigma, U', A)$ generates substrings of strings in $L(G)$ and has the property that for each $y \in L$ there exists $w \in \{2, 3\}^*$ such that $\$y[w]\$$ is a substring of some string in $L(G')$. From the closure properties of $E(K)$, see Theorem 2.1(2), it then easily follows that $L \in E(K)$. Roughly, A consists of all substrings of strings $b_1 \cdots b_m$ (see statement (P)), whereas U' is obtained from U by replacing each language of K by one of its strings. Property 4.3(ii) allows us to replace some of the U-derivations $b_i \overset{*}{\Rightarrow} v_i$ by U'-derivations, whereas statement (P) and property 4.3(i) ensure that sufficiently many of them can be so replaced, obtaining at least one $\$y[w]\$$ as a substring.

We now start the formal proof.

Part 1. Suppose that statement (P) is false. Thus, there exists a string $y_0 \in L$ such that for every cut$\langle w_1,..., w_n\rangle$ and every derivation in G of $\$w_1\$y_0[t(w_1)]\cdots\$w_n\$y_0[t(w_n)]$ the number of integers i with $\$ \in \text{alph}(v_i)$ is at most k in each decomposition (P*) of that derivation.

We shall construct from G a FIN-iteration grammar with a regular control language, generating CUT (cf. Theorem 2.1(3)). Before doing this we need, for technical reasons, two (more or less standard) transformations on G. We first change G into an equivalent K-iteration grammar $G_1 = (V_1, \Sigma, U_1, A_1)$ such that

each symbol in V_1 has an indication of the alph of the terminal string it generates. Let $V_1 = \{\langle a, \Omega\rangle \mid a \in V \text{ and } \Omega \subseteq \Sigma\}$. Henceforth we identify, for each $a \in \Sigma$, $\langle a, \{a\}\rangle$ with a; thus $\Sigma \subseteq V_1$. Let $A_1 = \{\langle S, \Omega\rangle \mid \Omega \subseteq \Sigma\}$. For each $f \in U$, U_1 contains the substitution f_1 such that $f_1(\langle a, \Omega\rangle) = \{\langle a_1, \Omega_1\rangle\langle a_2, \Omega_2\rangle \cdots \langle a_n, \Omega_n\rangle \mid a_1 a_2 \cdots a_n \in f(a)$ and $\Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_n = \Omega\}$, where $\Omega = \varnothing$ if $n = 0$. It is easy to show that the derivations of terminal strings in G_1 are those of G, in which each symbol is labeled with the alph of the terminal string it generates. Hence $L(G_1) = L(G)$.

Since we want, later, to erase all symbols $\langle a, \Omega\rangle$ such that $\Omega \cap \{\$, 0, 1\} = \varnothing$, we now ensure with the help of a control language that the corresponding transformation of G_1 will not produce any derivations which are not obtained from derivations of G_1. In fact, we want to ensure that from every sentential form of the new grammar a terminal string can be generated. We construct a K-iteration grammar $G_2 = (V_1, \Sigma, U_2, A_1)$ and a regular control language $M \subseteq U_2^*$ such that $L(G_2, M) = L(G_1)$. For each $f \in U_1$ and $W \subseteq V_1$, U_2 contains a substitution $\langle f, W\rangle$ such that, for $b \in V_1$, $\langle f, W\rangle(b) = f(b) \cap W^*$. The control language M consists of all strings $\langle f_1, W_1\rangle \cdots \langle f_m, W_m\rangle$ such that, for $1 \leqslant i \leqslant m$, $W_i = \{b \in V_1 \mid f_m(f_{m-1}(\cdots f_{i+1}(b)\cdots)) \cap \Sigma^* \neq \varnothing\}$, i.e., W_i is the set of all symbols that can generate some terminal string according to the rest of the control string. Since $W_m = \Sigma$ and $W_{i-1} = \{b \mid f_i(b) \cap W_i^* \neq \varnothing\}$, M can be recognized by a finite automaton, and so M is regular (cf. also Theorem 1 of [39]). Clearly $L(G_2, M) = L(G_1)$ and they have the same derivations of terminal strings. G_2 has the desired property that from every sentential form, obtained by a prefix of some control string in M, a terminal string can be generated by the rest of the control string.

We now change G_2 into a FIN-iteration grammar G_3 (with regular control language) that generates the language CUT. Let $G_3 = (V_3, \{\$, 0, 1\}, U_3, A_3)$, where $V_3 = \{\langle a, \Omega\rangle \in V_1 \mid \Omega \cap \{\$, 0, 1\} \neq \varnothing\}$, $A_3 = \{\langle S, \Omega\rangle \mid \Omega \cap \{\$, 0, 1\} \neq \varnothing\}$ and, for each $\langle f, W\rangle \in U_2$, U_3 contains a substitution $[f, W]$ such that, for each $b \in V_3$, $[f, W](b) = e(\langle f, W\rangle(b) \cap R)$, where $R = \{\langle a_1, \Omega_1\rangle \cdots \langle a_m, \Omega_m\rangle \mid \#(\{i \mid \$ \in \Omega_i, 1 \leqslant i \leqslant m\}) \leqslant k\}$ and e is the homomorphism such that $e(\langle a, \Omega\rangle) = \lambda$ if $\Omega \cap \{\$, 0, 1\} = \varnothing$ and $e(\langle a, \Omega\rangle) = \langle a, \Omega\rangle$ otherwise. G_3 is provided with the same control language M as G_2 (with square brackets instead of angular). It follows from the property of G_2 mentioned above, that each derivation of G_3 (using a control string from M) can be converted into a corresponding derivation of G_2. Hence $L(G_3, M) \subseteq e(L(G_2, M)) = \text{CUT}$. On the other hand, the fact that statement (P) is false, implies that $\text{CUT} \subseteq L(G_3, M)$. Hence $L(G_3, M) = \text{CUT}$. Note that if $\langle a, \Omega\rangle$ appears in a derivation of some terminal string in G_3, then $\Omega \cap \{\$, 0, 1\}$ is the alph of the terminal string it generates.

Finally, we change G_3 by removing from each language $g(b)$, with $g \in U_3$ and $b \in V_3$, all useless strings, i.e., strings w that are never used in a terminal derivation (controlled by M) of the form $S' \overset{*}{\Rightarrow} \cdots b \cdots \Rightarrow \cdots w \cdots \overset{*}{\Rightarrow} \cdots$, with $S' \in A_3$.

We now claim that G_3 is a FIN-iteration grammar. Obviously A_3 is finite. Suppose that $g(b)$ is infinite for some $g \in U_3$ and $b \in V_3$. Then there exist a sequence $\langle a_1, \Omega_1\rangle, \ldots, \langle a_m, \Omega_m\rangle$ with $\$ \in \Omega_i$ $(1 \leqslant i \leqslant m)$ and $0 \leqslant m \leqslant k$, and a subalphabet

$W \subseteq \{\langle a, \Omega\rangle \mid \$ \notin \Omega\}$, such that $g(b)$ contains infinitely many strings from the set $B = \{v_1\langle a_1, \Omega_1\rangle\, v_2\langle a_2, \Omega_2\rangle \cdots v_m\langle a_m, \Omega_m\rangle\, v_{m+1} \mid v_i \in W^*$ and $\text{alph}(v_1 v_2 \cdots v_{m+1}) = W\}$. Pick one string $w \in g(b) \cap B$ and consider a derivation in which w is used: $S' \overset{*}{\Rightarrow} \cdots b \cdots \Rightarrow \cdots w \cdots \overset{*}{\Rightarrow} u \in L(G_3, M) = \text{CUT}$. In this derivation we can replace w by any other string $w' \in g(b) \cap B$, using the given subderivations starting with the symbols of w to construct a derivation $S' \overset{*}{\Rightarrow} \cdots b \cdots \Rightarrow \cdots w' \cdots \overset{*}{\Rightarrow} u' \in \text{CUT}$. Since we can keep the subderivations starting with $\langle a_1, \Omega_1\rangle$,..., $\langle a_m, \Omega_m\rangle$ fixed, u' contains the same number of occurrences of \$ as u. Hence, by varying w', we would obtain an infinite number of strings u' in CUT (note that each $\langle a, \Omega\rangle \in V_3$ generates at least one symbol, because $\Omega \cap \{\$, 0, 1\} \neq \emptyset$) with the same number of \$-symbols, i.e., an infinite number of cuts with the same number of nodes. This contradicts (C3) of Section 3. Hence G_3 is a FIN-iteration grammar and consequently $L(G_3, M) \in H(\text{FIN}) = \text{ETOL}$, contradicting the fact that $\text{CUT} \notin \text{ETOL}$ (Corollary 3.3). Hence statement (P) is true.

Part 2. Since we want the axioms of our new grammar G' (cf. the introduction of this proof) to consist of substrings of sentential forms of G, we have to take care that the omitted context is not disregarded. Thus, we construct an ETOL scheme G_E and for each $V_1 \subseteq V$ we construct a regular control language $M(V_1)$ and a language of axioms $A(V_1) \in K$, such that from the union of the correspondingly generated languages, the language L can be obtained by an ngsm mapping. For any language L_0 over V and any subalphabet V_0 of V, let $w(L_0, V_0)$ denote a fixed string $w \in L_0$ such that $\text{alph}(w) \subseteq V_0$ (if it exists). We construct the ETOL scheme $G_E = (V, \Sigma, U_E)$ such that if $f \in U$, then $f_E \in U_E$, where, for every $a \in V$, $f_E(a) = \{w(f(a), V_0) \mid V_0 \subseteq V\}$. For given $V_1 \subseteq V$ we define the language $A(V_1)$ of axioms to consist of all nonempty strings $w \in V^*$ such that there is a derivation in G of the form $S \overset{*}{\Rightarrow} u_1' a u_2' \Rightarrow u_1 w_1 w w_2 u_2$ (with $u_1' \Rightarrow u_1$, $u_2' \Rightarrow u_2$, $a \Rightarrow w_1 w w_2$, and u_1', u_1, u_2', u_2, w_1, $w_2 \in V^*$) and $\text{alph}(u_1 w_1 w w_2 u_2) = V_1$. In other words, $A(V_1)$ consists of all substrings of strings in $f(a)$, with $f \in U$ and $a \in V$, which can occur in a sentential form with alph V_1. For given $a \in V$ and $f \in U$ there are of course a finite number of possible pairs $\langle \text{alph}(u_1), \text{alph}(u_2)\rangle$, where in the above derivation the last step is made by f. It should therefore be clear that an ngsm can single out from the strings of $f(a)$ all substrings which can "occur in alph V_1" (note that we do not care about constructivity). Since K is closed under ngsm mappings, $A(V_1) \in K$. For given $V_1 \subseteq V$ we define the control language $M(V_1) \subseteq U_E^*$ to consist of all $\pi \in U_E^*$ such that each $a \in V_1$ generates (in G) some terminal string under the control string $h_E(\pi)$, where h_E is the homomorphism such that $h_E(f_E) = f$. It follows from Theorem 1 of [39] that $M(V_1)$ is regular.

Let $G_E(V_1)$ denote the (K, FIN)-iteration grammar $(V, \Sigma, U_E, A(V_1))$. We note that, by construction, $L(G_E(V_1), M(V_1))$ contains substrings of strings of $L(G)$ only. In what follows we will show that for each $y \in L$ there exists $w \in \{0, 1\}^*$ such that $\$y[t(w)]\$$ is a substring of a string in $L(G_E(V_1), M(V_1))$ for some $V_1 \subseteq V$. And consequently, using the closure properties of $E(K)$ (in particular, closure under ngsm mappings in order to extract y from the string containing $\$y[t(w)]\$$) and the

fact that we can get rid of regular control (Theorem 2.1), $L \in E(K)$ and the theorem is proved.

Recall that $k = (\#(V)+1)^2 + \#(V)$. Consider an arbitrary $y \in L$. According to statement (P) there exist a cut $\langle w_1, \ldots, w_n \rangle$ and a derivation (P*) in G of the string $v = \$w_1 \$y[t(w_1)] \cdots \$w_n \$y[t(w_n)]$ such that $\#(\{i \mid \$ \in \text{alph}(v_i),\ 1 \leqslant i \leqslant m\}) > k$; see (P) for notation. The symbols which occur exactly once in $b_1 \cdots b_m$ divide the string $b_1 \cdots b_m$ into at most $\#(V)+1$ pieces. Hence, in at least one of these pieces there are more than $\#(V)+1$ symbols b_i that generate \$. Thus, there exist i_1 and i_2 $(1 \leqslant i_1 < i_2 \leqslant m)$ such that $\#(\{i \mid \$ \in \text{alph}(v_i),\ i_1 \leqslant i \leqslant i_2\}) > \#(V)+1$, and if $i_1 \leqslant i \leqslant i_2$ then b_i occurs at least twice in $b_1 \cdots b_m$. Therefore the string $v_{i_1} \cdots v_{i_2}$ generated by $b_{i_1} \cdots b_{i_2}$ has more than $\#(V)$ substrings of the form $\$w\$y[t(w)]\$$. Consider any symbol b_i $(i_1 \leqslant i \leqslant i_2)$ such that $v_i \in N$ (for the definition of N see Lemma 4.3). It follows from property 4.3(i) that all strings v_j with $b_j = b_i$ lie within the same substring $\$w\$y[t(w)]\$$ of v. Since $v_{i_1} \cdots v_{i_2}$ has more than $\#(V)$ such substrings, we conclude that there exist p_1 and p_2 $(1 \leqslant i_1 \leqslant p_1 < p_2 \leqslant i_2 \leqslant m)$ such that $v_{p_1} \cdots v_{p_2}$ has a substring of the form $\$w\$y[t(w)]\$$ and for all i, $p_1 \leqslant i \leqslant p_2$, $v_i \notin N$.

Thus we have a derivation $b_{p_1} \cdots b_{p_2} \overset{*}{\Rightarrow} v_{p_1} \cdots v_{p_2}$ such that each b_i generates a string not in N, and $v_{p_1} \cdots v_{p_2}$ has a substring of the form $\$y[t(w)]\$$. We now want to argue that $v_{p_1} \cdots v_{p_2} \in L(G_E(V_1), M(V_1))$, where $V_1 = \text{alph}(u_1' b_1 \cdots b_m u_2')$. Obviously $b_{p_1} \cdots b_{p_2} \in A(V_1)$. We will show, using property 4.3(ii), how to transform the derivation $b_{p_1} \cdots b_{p_2} \overset{*}{\Rightarrow} v_{p_1} \cdots v_{p_2}$ (without changing $b_{p_1} \cdots b_{p_2}$ or $v_{p_1} \cdots v_{p_2}$) into a derivation of $G_E(V_1)$, such that the new control string is obtained from the old control string by adding the subscript E to each substitution (and consequently it belongs to $M(V_1)$). Consider the first step of the derivation: $b_{p_1} \cdots b_{p_2} \Rightarrow u_{p_1} \cdots u_{p_2} \overset{*}{\Rightarrow} v_{p_1} \cdots v_{p_2}$ with $b_i \Rightarrow u_i \overset{*}{\Rightarrow} v_i$ and $u_i \in f(b_i)$, $f \in U$. For some i, we replace the subderivation $b_i \Rightarrow u_i \overset{*}{\Rightarrow} v_i$ by a derivation $b_i \Rightarrow w(f(b_i), \text{alph}(u_i)) \overset{*}{\Rightarrow} v_i$. The existence of such a derivation can be seen as follows. Let $u_i = c_1 \cdots c_q \overset{*}{\Rightarrow} x_1 \cdots x_q = v_i$ with $c_j \overset{*}{\Rightarrow} x_j$ $(c_j \in V,\ x_j \in V^*)$. Then, since $\text{alph}(w(f(b_i), \text{alph}(u_i))) \subseteq \text{alph}(u_i)$, we have that $w(f(b_i), \text{alph}(u_i)) = c_{j_1} c_{j_2} \cdots c_{j_s} \overset{*}{\Rightarrow} x_{j_1} x_{j_2} \cdots x_{j_s}$ (where $1 \leqslant j_r \leqslant q$ for all r, $1 \leqslant r \leqslant s$). Since $\text{alph}(x_{j_1} \cdots x_{j_s}) \subseteq \text{alph}(x_1 \cdots x_q)$, property 4.3(ii) implies that $x_{j_1} \cdots x_{j_s} = v_i$. Hence $w(f(b_i), \text{alph}(u_i)) \overset{*}{\Rightarrow} v_i$. By repeating this replacement for each i, $p_1 \leqslant i \leqslant p_2$, we obtain a derivation $b_{p_1} \cdots b_{p_2} \Rightarrow u'_{p_1} \cdots u'_{p_2} \overset{*}{\Rightarrow} v_{p_1} \cdots v_{p_2}$ such that for each i $(p_1 \leqslant i \leqslant p_2)$ there exists $V_0 \subseteq V$ with $u_i' = w(f(b_i), V_0)$. Thus $u_i' \in f_E(b_i)$, i.e., the first step of this derivation is obtained by the substitution f_E of G_E. We now note that each symbol in the string $u'_{p_1} \cdots u'_{p_2}$ generates a terminal string not in N (because it is a substring of some v_i) and consequently we can continue changing the first step of $u'_{p_1} \cdots u'_{p_2} \overset{*}{\Rightarrow} v_{p_1} \cdots v_{p_2}$ in the same way as we did for $b_{p_1} \cdots b_{p_2} \overset{*}{\Rightarrow} v_{p_1} \cdots v_{p_2}$. After changing all steps of the derivation in this fashion we end up with the desired derivation in $G_E(V_1)$ with control string in $M(V_1)$. This shows that $L(G_E(V_1), M(V_1))$ contains a string that has a substring of the form $\$y[t(w)]\$$, and the proof is completed. ∎

Theorem 4.4 suffices to show that the "basic hierarchy" $\{B^n(\text{FIN})\}_{n \geqslant 1}$ is proper.

4.5. THEOREM. *For every* $n \geqslant 1$, $B^n(\text{FIN}) \subsetneq H(B^n(\text{FIN})) \subsetneq B^{n+1}(\text{FIN})$.

Proof. By Corollary 3.9 of [21], $B(\mathrm{FIN})$ is not closed under substitution and so $B(\mathrm{FIN}) \subsetneq H(B(\mathrm{FIN}))$. Now assume that $B^n(\mathrm{FIN}) \subsetneq H(B^n(\mathrm{FIN}))$. Let $L \in H(B^n(\mathrm{FIN})) - B^n(\mathrm{FIN})$. Since $B^{n+1}(\mathrm{FIN})$ is closed under cut (Lemma 4.2), $\mathrm{cut}(L) \in B^{n+1}(\mathrm{FIN})$. By Theorem 4.4, if $\mathrm{cut}(L) \in H(B^n(\mathrm{FIN}))$, then $L \in E(B^n(\mathrm{FIN}))$, and hence $L \in B^n(\mathrm{FIN})$, because $B^n(\mathrm{FIN})$ is closed under iterated finite substitution (Theorem 2.2(2)). Consequently $\mathrm{cut}(L) \notin H(B^n(\mathrm{FIN}))$, and so $H(B^n(\mathrm{FIN})) \subsetneq B^{n+1}(\mathrm{FIN})$. Now consider $L' = \tau(L_1, \mathrm{cut}(L))$ where L_1 is any language in $B(\mathrm{FIN}) - \mathrm{CF}$. Clearly $L' \in H(B^{n+1}(\mathrm{FIN}))$. By Theorem 2.2(4) and (1), if $L' \in B^{n+1}(\mathrm{FIN})$ then $\mathrm{cut}(L) \in H(B^n(\mathrm{FIN}))$. Hence $B^{n+1}(\mathrm{FIN}) \subsetneq H(B^{n+1}(\mathrm{FIN}))$. ∎

4.6. COROLLARY. *The smallest full basic-*AFL $B^*(\mathrm{FIN})$ *is the union of an infinite hierarchy of full principal hyper-*AFL*s. In particular* $B^*(\mathrm{FIN})$ *is not full principal.*

Proof. $B^*(\mathrm{FIN})$ is the union of the proper hierarchy of full hyper-AFLs $\{H(B^n(\mathrm{FIN}))\}$. $B(\mathrm{FIN})$ is full principal (see [18]) and by Corollary 4.9 of [21] the operators B and H preserve full principality. Thus every $B^n(\mathrm{FIN})$ and $H(B^n(\mathrm{FIN}))$ is full principal. ∎

Theorem 4.5 also implies that the bounded nested stack automata form an infinite hierarchy with respect to the depth of nesting of stacks (Proposition 5.7 of [21]).

5. GENERAL HIERARCHY RESULTS

It follows from the results in Section 4 that the operator B is hierarchical for every full semi-AFL K that is closed under iterated finite substitution: if $K \subsetneq B(K)$ then $B^n(K) \subsetneq B^{n+1}(K)$ for all $n \geqslant 1$ (cf. the proof of Theorem 4.5). We now want to show that B is hierarchical for all full semi-AFLs. To do this it suffices to prove that if $K \subsetneq E(K)$ then $E(K) \subsetneq H(K)$. In the next theorem we show that if $K \subsetneq E(K)$ then $E(K)$ is not closed under substitution. The theorem is a generalization of the syntactic lemma of [26]. For the syntactic substitution τ, see Section 1.

5.1. THEOREM. *Let K be a full semi-*AFL, *and let L_1, L_2 be two languages. If $\tau(L_1, L_2) \in E(K)$, then $L_1 \in K$ or $L_2 \in$* ETOL.

Proof. The proof is similar to both that of Theorem 4.4 and that of the syntactic lemma (Lemma 2.1 of [26]). By the closure properties of K and ETOL we may assume that $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ with $\Sigma_1 \cap \Sigma_2 = \varnothing$. Let $G = (V, \Sigma, U, A)$ be a (K, FIN)-iteration grammar generating $\tau(L_1, L_2)$; thus $\Sigma = \Sigma_1 \cup \Sigma_2$. We consider two cases.

Case 1. For each string $a_1 a_2 \cdots a_n \in L_1$ (with $a_i \in \Sigma_1$) there exist $y \in L_2$ and a derivation $b_1 \cdots b_k \overset{*}{\Rightarrow} v_1 \cdots v_k = a_1 y a_2 y \cdots a_n y$ of G (with $b_1 \cdots b_k \in A$, $b_i \in V$, and $b_i \overset{*}{\Rightarrow} v_i$) such that, for all i $(1 \leqslant i \leqslant k)$, $v_i \in \Sigma_2^* \cup \Sigma_2^* \Sigma_1 \Sigma_2^*$, i.e., v_i contains at most

one occurrence of a symbol from Σ_1. In this case we want to show that $L_1 \in K$. Consider the set Φ of all finite substitutions $\phi: V \to \Sigma_1^*$ such that

(i) for every $b \in V$, $\phi(b) \subseteq \Sigma_1 \cup \{\lambda\}$

(ii) there exists $\pi \in U^*$ such that for every $b \in V$

(1) if $a \in \phi(b)$ with $a \in \Sigma_1$, then $b \Rightarrow^\pi v$ for some $v \in \Sigma_2^* a \Sigma_2^*$, and

(2) if $\lambda \in \phi(b)$, then $b \Rightarrow^\pi v$ for some $v \in \Sigma_2^*$.

Clearly, by (i), Φ is finite. It is easy to see, by the assumption of this case, that $L_1 = \bigcup \{\phi(A) \mid \phi \in \Phi\}$. Thus, since $A \in K$, $L_1 \in K$.

Case 2. We now assume the negation of Case 1. This *implies* that there exists a string $a_1 a_2 \cdots a_n \in L_1$ such that for every $y \in L_2$ there is a derivation $b_1 \cdots b_k \overset{*}{\Rightarrow} v_1 \cdots v_k = a_1 y a_2 y \cdots a_n y$ with $v_i \in \Sigma^* \Sigma_1 y \Sigma_1 \Sigma^*$ for some i, $1 \leqslant i \leqslant k$.

We now show that $L_2 \in \mathrm{ETOL}$. For every $V_1 \subseteq V$ such that $V_1 = \mathrm{alph}(w)$ for some $w \in A$, let $G(V_1)$ be the FIN-iteration grammar (V, Σ, U, V_1) and let $M(V_1) = \{\pi \in U^* \mid$ for every $b \in V_1$ there exists $w \in \Sigma^*$ such that $b \Rightarrow^\pi w\}$. By Theorem 1 of [39], $M(V_1)$ is regular. It should be clear that each $L(G(V_1), M(V_1))$ contains substrings of strings of $L(G)$ only. Moreover, due to the assumption of this case, for every $y \in L_2$ there is a string $w \in L(G(V_1), M(V_1))$ for some V_1, such that $w \in \Sigma^* \Sigma_1 y \Sigma_1 \Sigma^*$. Consequently L_2 can be obtained from the union of all $L(G(V_1), M(V_1))$ by an ngsm which extracts y from every string containing aya' for some $a, a' \in \Sigma_1$. Hence, since ETOL is a full semi-AFL, $L_2 \in \mathrm{ETOL}$. ∎

With exactly the same proof as that of Theorem 5.1 one can prove: if K_1 and K_2 are full semi-AFLs and $\tau(L_1, L_2) \in H(K_1, K_2)$, then $L_1 \in K_1$ or $L_2 \in H(K_2)$.

Combining Theorems 5.1, 4.4, and 2.2(4) we define an operation on languages, called "subcut," which "cracks" the B-operator. The subcut operation combines the cut operation with the operation of (syntactic) substitution, using the language CUT. Recall that the operation $\tau(L_1, L_2)$ involves an implicit change of alphabet.

5.2. Definition. For every language L, $\mathrm{subcut}(L) = \tau(\mathrm{CUT}, \mathrm{cut}(\tau(L, \mathrm{CUT})))$.

Note that every full basic-AFL K is closed under subcut: $\mathrm{CUT} \in B(\mathrm{ONE}) \subseteq K$, and K is closed under cut (Lemma 4.2) and under substitution.

5.3. Theorem. *Let K be a full semi-*AFL *and L a language. If* $\mathrm{subcut}(L) \in B(K)$, *then* $L \in K$.

Proof. Let $\mathrm{subcut}(L) \in B(K)$. By Theorem 2.2(4), $\mathrm{cut}(\tau(L, \mathrm{CUT})) \in H(K)$, because $\mathrm{CUT} \notin \mathrm{CF} \subseteq \mathrm{ETOL}$ (Corollary 3.3). Hence, by Theorem 4.4, $\tau(L, \mathrm{CUT}) \in E(K)$. Finally, by Theorem 5.1, $L \in K$, because $\mathrm{CUT} \notin \mathrm{ETOL}$ (Corollary 3.3). ∎

5.4. Corollary. *Let K be a full semi-*AFL. *The following statements are equivalent:*

(1) *K is a full basic-*AFL.

(2) *$B(K)$ is a full basic-*AFL.

(3) *$B(K)$ is closed under substitution.*

(4) $H(K) = B(K)$.

(5) *$H(K)$ is a full basic-*AFL.

Proof. (1) $\Rightarrow$ (2) $\Rightarrow$ (3), (1) $\Rightarrow$ (4) $\Rightarrow$ (3), and (1) $\Rightarrow$ (5) are obvious. To prove (3) $\Rightarrow$ (1) consider $L \in B(K)$. Since $B(K)$ is substitution closed, subcut$(L) \in B(K)$. Hence, by Theorem 5.3, $L \in K$. Thus $B(K) \subseteq K$, i.e., K is a full basic-AFL. To prove (5) $\Rightarrow$ (4) note that, by Theorem 3.1 of [21], $B(K) = B(H(K))$. Hence, since $H(K)$ is a full basic-AFL, $B(K) = H(K)$. ▮

The next theorem is the main result of this paper. It shows that B is hierarchical.

5.5. THEOREM. *Let K be a full semi-*AFL. *If $K \subsetneq B(K)$, then, for every $n \geqslant 1$, $B^n(K) \subsetneq H(B^n(K)) \subsetneq B^{n+1}(K)$, $B^n(K)$ is not substitution closed, and $B^*(K)$ is the union of an infinite hierarchy of full hyper-*AFL*s containing K.*

Proof. It is immediate from (1), (2), and (3) of Corollary 5.4 that, for every $n \geqslant 1$, $B^n(K) \subsetneq B^{n+1}(K)$, and $B^n(K)$ is not substitution closed. Since $H(B^n(K))$ is closed under substitution, it lies properly between $B^n(K)$ and $B^{n+1}(K)$. Finally, note that $B^*(K)$ is the union of the full hyper-AFLs $H(B^n(K))$. ▮

It is shown in Theorem 4.8 of [27] for any full AFL K_1 that if K_1 is not substitution closed, then $K_1 \subsetneq \hat{F}_\sigma(K_1) \subsetneq A(K_1)$, where $\hat{F}_\sigma(K_1)$ is the smallest substitution-closed full AFL containing K_1, and $A(K_1)$ is the smallest full super-AFL containing K_1 (see Sect. 2). Hence, for $K_1 = B^n(K)$ as in Theorem 5.5, $K_1 \subsetneq \hat{F}_\sigma(K_1) \subsetneq A(K_1) \subseteq H(K_1) \subsetneq B(K_1)$. As an intermezzo we will show that the inclusion $A(K_1) \subseteq H(K_1)$ is also proper. For the operation f used in the next lemma, cf. Theorem 5.2 of [16].

5.6. LEMMA. *Let K be a full semi-*AFL. *Let L be a language over the alphabet Σ and let $f(L) = \{wa_1wa_2w \cdots wa_nw \mid a_i \in \Sigma,\ a_1a_2 \cdots a_n \in L,\ w \in b^*\}$, where $b \notin \Sigma$. If $f(L) \in A(K)$, then $L \in K$.*

Proof. Let $f(L) \in A(K)$. Then there exists a (generalized) context-free grammar $G = (V, N, \Sigma_b, P, S)$ such that $L(G) = f(L)$, where Σ_b is the finite set of terminals ($\Sigma_b = \Sigma \cup \{b\}$), N is the finite set of nonterminals ($N \cap \Sigma_b = \varnothing$), $V = N \cup \Sigma_b$, $S \in N$ is the initial nonterminal, and P is an *infinite* set of context-free productions of the form $D \to y$ with $D \in N$ and $y \in V^*$, such that for every $D \in N$ the set $rhs(D) = \{y \in V^* \mid D \to y \text{ is in } P\}$ is in K. In other words, G is a K-extended context-free grammar, cf. [36, 38, 5]; see also Theorem 4.5 of [27]. For $D \in N$, let $L(D) = \{y \in \Sigma_b^* \mid D \overset{*}{\Rightarrow} y\}$, and for $c \in \Sigma_b$, let $L(c) = \{c\}$. We may assume that $L(D) \neq \varnothing$ for all $D \in N$. For $D \in N$, let $b(D) = \{w \in b^* \mid \text{there exists } y \in L(D) \text{ such that } y \text{ has a substring } a_1wa_2 \text{ for some } a_1, a_2 \in \Sigma\}$. We now consider three subalphabets of V. Let

$V_0 = \{d \in V \mid L(d) \subseteq b^*\}$, $V_1 = \{d \in V \mid L(d) \subseteq b^* \cup b^*\Sigma b^*\} \cup \{D \in N \mid b(D)$ is a singleton$\}$, and $V_2 = \{D \in N \mid \#(b(D)) \geq 2\}$. Thus $V_0 \subseteq V_1$, and $V_1 \cap V_2 = \varnothing$, $V_1 \cup V_2 = V$.

Suppose $D \in V_2$ is used in a derivation of G. Then the other symbols in the sentential form in which D occurs cannot generate an element of Σ: otherwise they would uniquely determine the string $w \in b^*$ of the generated terminal string $wa_1wa_2w \cdots wa_nw$, contradicting the fact that $\#(b(D)) \geq 2$. Hence they are all in V_0. We now construct another K-extended context-free grammar $G_1 = (V, N, \Sigma_b, P_1, S)$ by intersecting, for every $D \in N$, $rhs(D)$ with the regular language $V_0^* V_2 V_0^* \cup V_1^*$. Then $L(G_1) = L(G)$.

Finally we construct a K-extended context-free grammar $G_2 = (V, N, \Sigma, P_2, S)$ such that $L(G_2) = L$. P_2 is obtained from P_1 by applying to every $rhs(D)$, $D \in N$, the finite substitution ϕ defined as follows. Let h_b be the homomorphism erasing b, i.e., $h_b(b) = \lambda$ and $h_b(a) = a$ for all $a \in \Sigma$. For every $d \in V_1$, $\phi(d) = h_b(L(d) \cap (b^* \cup b^*\Sigma b^*))$, and for every $D \in V_2$, $\phi(D) = \{D\}$. Note that for $d \in V_0$, $\phi(d) = \{\lambda\}$. Clearly $L(G_2) \subseteq L$. To see that $L \subseteq L(G_2)$,. consider $a_1 \cdots a_n \in L$, and take $w \in b^*$ such that for no $D \in N$, $b(D) = \{w\}$. Then the symbols $d \in V_1$ occurring in the sentential forms of a derivation of $wa_1wa_2 \cdots a_nw$ generate strings from $L(d) \cap (b^* \cup b^*\Sigma b^*)$ only. Hence $L(G_2) = L$.

From the construction of G_1 and G_2 it should be clear that in G_2, for every $D \in N$, $rhs(D) \subseteq N \cup \Sigma^*$ (i.e., productions in G_2 are of the form $D_1 \to D_2$ or $D_1 \to y$ with $D_1, D_2 \in N$ and $y \in \Sigma^*$). Hence $L(G_2)$ is the union of finitely many languages of the form $rhs(D) \cap \Sigma^*$, $D \in N$. Consequently $L(G_2) \in K$, and so $L \in K$. ∎

5.7. COROLLARY. *Let K be a full semi-*AFL. *K is a full hyper-*AFL *if and only if $A(K) = H(K)$.*

Proof. It is easy to see that every full hyper-AFL is closed under the operation f of Lemma 5.6 (in fact closure under E suffices). Suppose $A(K) = H(K)$ and consider $L \in H(K)$. Then $f(L) \in H(K) = A(K)$ and hence, by Lemma 5.6, $L \in K$. Thus $H(K) \subseteq K$, i.e., K is a full hyper-AFL. ∎

5.8. COROLLARY. *Let K be a full semi-*AFL *which is not closed under substitution. Then $K \subsetneq \hat{F}_\sigma(K) \subsetneq A(K) \subsetneq H(K) \subsetneq B(K)$.*

Proof. $K \subsetneq \hat{F}_\sigma(K) \subsetneq A(K)$ by Theorem 4.8 of [27] (note that a full semi-AFL is substitution closed if and only if $\hat{F}(K)$ is substitution closed [26]). By Corollary 5.7, $A(K) \subsetneq H(K)$, and by Corollary 5.4(1) and (4), $H(K) \subsetneq B(K)$. ∎

After this intermezzo on full super-AFLs we return to the properties of the B-operator. The existence of the subcut operation with the following two properties (for every full semi-AFL K):

(i) if subcut$(L) \in B(K)$ then $L \in K$ (Theorem 5.3),

(ii) if $B(K) \subseteq K$ then K is closed under subcut,

ensures that the B-operator has most of the nice properties of a syntactic operator,

as defined in [28] (of which substitution was the first example, see [26]). Actually, the above two properties do not imply that B is a syntactic operator for the following two reasons. Firstly, syntactic operators are always binary whereas B is unary. This is not a real problem: a theory of unary syntactic operators could easily be developed analogous to the binary case (cf. [30]). Secondly, (ii) is much weaker than the requirement for syntactic operators which says (in this case) that $B(K)$ should be equal to $\hat{S}(\{\text{subcut}(L) \mid L \in K\})$. We conjecture that the latter requirement is not true. However, it is not difficult to see that most of the results for syntactic operators also hold when the general version of property (ii) is used instead (together with some other elementary properties of a syntactic operator: it should be extensive, monotonic, and preserve the property of being a full semi-AFL). In fact, this would also slightly simplify the theory: the notion of "Ω-hierarchical" (Definition 2.6 of [28]) would become superfluous. Finally, the properties of B are also stronger than those of a syntactic operator, because the role played by full AFLs for syntactic operators is taken over by full hyper-AFLs for B (i.e., in results for syntactic operators, $\hat{F}$ should be replaced by H, or better $\hat{H}$, to obtain a possible result for B). As an example, in Corollary 5.4, (1)$\Leftrightarrow$(2) is analogous to Lemma 2.3 of [28] and Theorem 3.1 of [26], and (1)$\Leftrightarrow$(5) is analogous to Theorem 2.1 of [28] and Corollary 1 of [26]; (1)$\Leftrightarrow$(3) is specific for B, it is stronger than the usual property ((1)$\Leftrightarrow$(2)) of syntactic operators. Theorem 5.5 is similar to Theorem 3.2 of [26] and Theorem 2.2 of [28].

In the remainder of this section we show some other properties of the B-operator, similar to those of syntactic operators. In particular we consider properties of full basic-AFLs. Note that in the next theorem B can be replaced by $\hat{H}$ (where $\hat{H}(K)$ is the smallest full hyper-AFL containing K).

5.9. THEOREM. *Let K be a full basic-*AFL. *Let K_1 and K_2 be full semi-*AFLs, *and let L be a language*:

(1) *If $K \subseteq B(K_1 \cup K_2)$ then $K \subseteq K_1$ or $K \subseteq K_2$.*

(2) *If $K \subseteq B(K_1)$ then $K \subseteq K_1$.*

(3) *If $K_1 \subsetneq K$ then $B(K_1) \subsetneq K$.*

(4) *If $K \subseteq \hat{H}(L)$ then $K \subseteq \hat{S}(L)$.*

Proof. (1) (See Theorem 2.3 of [24] which says that every substitution closed full AFL is "fully prime".) Let $K \subseteq B(K_1 \cup K_2)$. Suppose that $K \not\subseteq K_1$ and $K \not\subseteq K_2$, and let $L_1 \in K - K_1$ and $L_2 \in K - K_2$. Since K is a full basic-AFL, the language $\text{subcut}(\tau(L_1, L_2))$ is in K. Hence $\text{subcut}(\tau(L_1, L_2)) \in B(K_1 \cup K_2) = B(\hat{S}(K_1 \cup K_2))$. Then, by Theorem 5.3, $\tau(L_1, L_2) \in \hat{S}(K_1 \cup K_2) \subseteq \text{Sub}(K_1, K_2)$. Hence, by the syntactic lemma for substitution (Lemma 2.1 of [26]), $L_1 \in K_1$ or $L_2 \in K_2$, contradicting the choice of L_1 and L_2. Clearly (1) implies (2), and (2) implies both (3) and (4). ∎

Theorem 5.9. shows that the operation of iterated substitution is of no effect in generating a full basic-AFL (e.g., (3) says that if $K_1 \subsetneq K$ then $H(K_1) \subsetneq K$; this

result is mentioned under point (1) in the introduction). One might say that every full basic-AFL K is "uniformly hyper-closed" in the sense that for an arbitrary family K_1 of languages, if $K = \hat{H}(K_1)$ then $K = \hat{S}(K_1)$, which follows easily from (2). This corresponds to Theorem 2.1 of [28] which says that every full AFL closed under any syntactic operator is "uniformly star-closed" (with $\hat{F}$ instead of $\hat{H}$).

We now turn to full principal basic-AFLs, i.e., full basic-AFLs which are full principal AFLs. Note that INDEX is a full principal basic-AFL. B^*(FIN) is not full principal (Corollary 4.6). For the next definition, see [28, 24, 9].

5.10. DEFINITION. Let K be a full AFL. The set of *nongenerators* of K is $N(K) = \{L \in K \mid \hat{F}(L) \subsetneq K\}$. The *exterior* of K is $\mathrm{Ext}(K) = \{L \mid K \not\subseteq \hat{F}(L)\}$.

Note that $N(K) = \mathrm{Ext}(K) \cap K$. If K is a substitution-closed full AFL (or closed under any other syntactic operator), then the $\hat{F}$ in the definition of $N(K)$ and $\mathrm{Ext}(K)$ can be replaced by $\hat{S}$ (Theorem 2.1 of [28]). If K is a full basic-AFL, then $\hat{F}$ can be replaced by $\hat{H}$, according to Theorem 5.9(4). It is well known that for a principal substitution-closed full AFL $K \neq \mathrm{REG}$ both $N(K)$ and $\mathrm{Ext}(K)$ are substitution-closed full AFLs and, moreover, $N(K)$ is the largest full semi-AFL properly contained in K, and $\mathrm{Ext}(K)$ is the largest full semi-AFL incomparable with K (see Theorem 4.4 of [26], Theorem 3.3 of [28], and [24]). This result holds in particular for full principal basic-AFLs, but for them we can show more (analogous to Theorem 3.3 of [28]).

5.11. THEOREM. *If K is a full principal basic-*AFL, *then $N(K)$ and* $\mathrm{Ext}(K)$ *are full basic-*AFLs.

Proof. Since $N(K) = \mathrm{Ext}(K) \cap K$, it suffices to show that $\mathrm{Ext}(K)$ is a full basic-AFL. Since K is full principal, there exists $L_0 \in K - \mathrm{Ext}(K)$. Hence $\mathrm{subcut}(L_0)$ is in K (because K is a full basic-AFL), but not in $B(\mathrm{Ext}(K))$ by Theorem 5.3. Thus $K \not\subseteq B(\mathrm{Ext}(K))$. This implies that $B(\mathrm{Ext}(K)) \subseteq \mathrm{Ext}(K)$: if $L \in B(\mathrm{Ext}(K))$ then $K \not\subseteq \hat{F}(L) \subseteq B(\mathrm{Ext}(K))$, and so $L \in \mathrm{Ext}(K)$. Hence $\mathrm{Ext}(K)$ is a full basic-AFL. ∎

Thus every full principal basic-AFL K has a largest full basic-AFL properly contained in K. This holds in particular for INDEX: INDEX has a largest proper full basic-AFL (hyper-AFL, super-AFL), cf. point (1) in the Introduction.

5.12. COROLLARY. N(INDEX) *is a full basic-*AFL.

Theorem 5.11 expresses the fact that for a full principal basic-AFL K the B-operator does not help generating K. Other ways of expressing the same fact are stated in the next theorem (cf. Theorem 5.9). Note that $\hat{B}(K)$ denotes the smallest full basic-AFL containing K.

5.13. THEOREM. *Let K be a full principal basic-*AFL. *Let K_1 and K_2 be full semi-*AFL*s, and let L be a language*:

(1) *If* $K \subseteq \hat{B}(K_1 \cup K_2)$ *then* $K \subseteq K_1$ *or* $K \subseteq K_2$.

(2) *If* $K \subseteq \hat{B}(K_1)$ *then* $K \subseteq K_1$.

(3) *If* $K_1 \subsetneq K$ *then* $\hat{B}(K_1) \subsetneq K$.

(4) *If* $K \subseteq \hat{B}(L)$ *then* $K \subseteq \hat{S}(L)$.

Proof. (1) If both K_1 and K_2 are contained in Ext(K), then $\hat{B}(K_1 \cup K_2) \subseteq \text{Ext}(K)$ by Theorem 5.11, and so $K \subseteq \text{Ext}(K)$, contradicting the fact that K is full principal. Hence K_i is not contained in Ext(K) for some $i = 1, 2$. Let $L \in K_i - \text{Ext}(K)$. Then $K \subseteq \hat{S}(L) \subseteq K_i$.

Clearly (1) implies (2), and (2) implies both (3) and (4). ∎

For (4) of Theorem 5.13, see Corollary 3 of Section 2 of [28]. It means that, for a full principal basic-AFL K, $\hat{F}$ can be replaced by $\hat{B}$ in the definition of $N(K)$ and Ext(K).

We finally note that, for a full principal basic-AFL K, iterated substitution does not help in generating K in the following strong sense (cf. point (2) of the introduction).

5.14. COROLLARY. *Let K be a full principal basic-*AFL. *If K_0 is a full semi-*AFL *properly contained in K and K_0 is not a full basic-*AFL, *then there exists an infinite hierarchy $\{K_n\}_{n \geq 1}$ of full hyper-*AFL*s such that $K_0 \subsetneq K_1 \subsetneq K_2 \subsetneq \cdots \subsetneq \bigcup_n K_n \subsetneq K$.*

Proof. Immediate from Theorem 5.5 and Theorem 5.11. ∎

6. HIERARCHIES INSIDE INDEX

From the previous sections we know two full basic-AFLs properly contained in INDEX: the smallest (B^*(FIN)) and the largest (N(INDEX)). In this section we show that they are different, that there is a full basic-AFL which lies properly in between them, and that these three full basic-AFLs are in fact separated by two infinite hierarchies of full hyper-AFLs.

Let us first show that $B^*(\text{FIN}) \subsetneq N(\text{INDEX})$. From Corollary 7.1 of [20] and Theorem 3.12 of [21] we know that if $L \in B(\text{ONE}) - \text{EDTOL}$ (recall that EDTOL = H(ONE)), then $L_1 = \{w \# w^R \mid w \in L\} \in \text{INDEX} - B^*(\text{FIN})$. Hence, e.g., $\{w \# w^R \mid w \in \text{CUT}\}$ is in INDEX but not in B^*(FIN). To show that L_1 is even in N(INDEX) we need a result from [29] saying that *2dgsm* mappings do not help in generating a substitution-closed full principal AFL. A *2dgsm* mapping is a mapping realized by a 2-way deterministic finite state transducer (for a definition, see, e.g., [3, 30, 16]).

6.1. DEFINITION. For a family K of languages, $2DGSM(K) = \{f(L) \mid L \in K$ and f is a *2dgsm* mapping$\}$.

In [29] $2DGSM(K)$ is denoted by FINITEVISIT(K), and in [16] by $DCS(K)$. It is well known (see, e.g., Theorem 2.10 of [29]) that if K is a full {semi-}AFL, then so is $2DGSM(K)$. The next proposition is Lemma 4.24 of [29].

6.2. PROPOSITION. *Let K be a substitution-closed full principal* AFL, *and let K_1 be a full semi-*AFL. *If $K \subseteq 2DGSM(K_1)$, then $K \subseteq K_1$.*

In other words, Ext(K) is closed under *2dgsm* mappings. Proposition 6.2 can be generalized to K_1-controlled EDTOL systems, see Theorem 3.2.17 of [16].

We can show now that the language L_1 above is in N(INDEX). Assume that $\hat{F}(L_1) = \text{INDEX}$. Since L_1 can be obtained from L by a *2dgsm* mapping, $L_1 \in 2DGSM(B(\text{FIN}))$. Hence $\text{INDEX} = \hat{F}(L_1) \subseteq 2DGSM(B(\text{FIN}))$. Thus, by Proposition 6.2, $\text{INDEX} \subseteq B(\text{FIN})$ which is a contradiction. Hence $L_1 \in N(\text{INDEX}) - B^*(\text{FIN})$, and so $B^*(\text{FIN}) \subsetneq N(\text{INDEX})$. To refine this result we consider two particular families of languages.

6.3. DEFINITION. $D_{CF} = 2DGSM(\text{CF}) \cap \text{INDEX}$ and $D_{EB} = 2DGSM(\text{EB}) \cap \text{INDEX}$, where $\text{EB} = B(\text{FIN})$.

Both D_{CF} and D_{EB} are full AFLs, and $D_{CF} \subseteq D_{EB}$. By Proposition 6.2, $D_{CF} \subseteq D_{EB} \subseteq N(\text{INDEX})$. In fact, as above, if $\text{INDEX} = \hat{F}(L)$ for some $L \in 2DGSM(B(\text{FIN}))$, then $\text{INDEX} \subseteq 2DGSM(B(\text{FIN}))$ and hence $\text{INDEX} \subseteq B(\text{FIN})$.

Thus, by Corollary 5.12, $B^*(\text{FIN}) \subseteq B^*(D_{CF}) \subseteq B^*(D_{EB}) \subseteq N(\text{INDEX})$. We will show that the first two inclusions are proper, that $B^*(D_{CF})$ is the union of an infinite hierarchy of full hyper-AFLs containing $B^*(\text{FIN})$, and that $B^*(D_{EB})$ is the union of an infinite hierarchy of full hyper-AFLs containing $B^*(D_{CF})$. From this it follows that $\text{REG} \subsetneq \hat{B}(\text{FIN}) \subsetneq \hat{B}(D_{CF}) \subsetneq N(\text{INDEX})$, separated by infinite hierarchies of full hyper-AFLs (cf. point (3) of the Introduction). To prove this we need a generalization of Corollary 7.1 of [20]: $\{w \# w^R \mid w \in L_1\} \in B^*(K)$, then $\{w \# w^R \mid w \in L_1\} \in H(K)$. We will only need the fact that $L = \{w \# w^R \mid w \in L_1\}$ has the following property (P2), see [19].

Property (P2). For all strings u, u', v, v', x, y, z over the alphabet of L, if $xuyvz$, $xu'yvz$, $xuyv'z$, and $xu'yv'z \in L$, then $u = u'$ or $v = v'$.

As shown in the next lemma, property (P2) forces each K-extended basic macro grammar the language of which is used in an iteration grammar generating L, to be linear (see Sect. 5 of [14]).

6.4. LEMMA. *Let K be a full semi-*AFL, *and let L be a language with property* (P2). *If $L \in H(B(K))$, then $L \in H(K)$.*

Proof. Let $G = (V, \Sigma, U, \{S\})$ be a $B(K)$-iteration grammar with $S \in V$, such that $L(G) = L$. Consider arbitrary $a \in V$ and $f \in U$. Let $G_{fa} = (F, \Psi, V, X, S_{fa}, d, P)$ be a K-extended basic macro grammar such that $L(G_{fa}) = f(a)$. We want to show that $f(a)$ can be changed into $L(G'_{fa})$ for some K-extended *linear* basic macro gram-

mar G'_{fa}, without changing $L(G)$. By doing this for every $f(a)$, G is turned into an $LB(K)$-iteration grammar generating L, and consequently $L \in H(LB(K))$. Since $LB(K) = H(K)$ by Theorem 2.2(1) and $H(K)$ is a full hyper-AFL, it follows that $L \in H(K)$ which proves the lemma.

To see that G_{fa} can be turned into a linear grammar we first note that $f(a)$ has the following property similar to property (P2). For all strings $u, u', v, v', x, y, z \in V^*$, if $xuyvz$, $xu'yvz$, $xuyv'z$, and $xu'yv'z$ are in $f(a)$, $\text{alph}(u) = \text{alph}(u')$ and $\text{alph}(v) = \text{alph}(v')$, then: (for all $w_1, w_2, w'_1, w'_2 \in V^*$ and $w \in \Sigma^*$) if $S \overset{*}{\Rightarrow} w_1 a w_2 \Rightarrow w'_1 xuyvzw'_2 \overset{*}{\Rightarrow} w$ in G, then either $S \overset{*}{\Rightarrow} w_1 a w_2 \Rightarrow w'_1 xu'yvzw'_2 \overset{*}{\Rightarrow} w$ or $S \overset{*}{\Rightarrow} w_1 a w_2 \Rightarrow w'_1 xuyv'zw'_2 \overset{*}{\Rightarrow} w$. In other words, under the given conditions, if $xuyvz$ is used in a derivation of a terminal string w, then either $xu'yvz$ or $xuyv'z$ can also be used to derive w. To see the truth of this property, consider a derivation $S \overset{*}{\Rightarrow} w_1 a w_2 \Rightarrow w'_1 xuyvzw'_2 \overset{*}{\Rightarrow} w$, see Fig. 2. Clearly, w can be written as $x_1 u_1 y_1 v_1 z_1$, where $w'_1 x \overset{*}{\Rightarrow} x_1$, $u \overset{*}{\Rightarrow} u_1$, $y \overset{*}{\Rightarrow} y_1$, $v \overset{*}{\Rightarrow} v_1$, and $zw'_2 \overset{*}{\Rightarrow} z_1$. Since $xu'yvz \in f(a)$ and $\text{alph}(u') = \text{alph}(u)$, we can replace, in this derivation, the subderivation $u \overset{*}{\Rightarrow} u_1$ by a subderivation $u' \overset{*}{\Rightarrow} u'_1$ for some $u'_1 \in \Sigma^*$, and thus $x_1 u'_1 y_1 v_1 z_1 \in L$. Similarly we can replace $v \overset{*}{\Rightarrow} v_1$ by $v' \overset{*}{\Rightarrow} v'_1 \in \Sigma^*$, and so $x_1 u_1 y_1 v'_1 z_1 \in L$ and also $x_1 u'_1 y_1 v'_1 z_1 \in L$. Since L has property (P2), $u_1 = u'_1$ or $v_1 = v'_1$. In case $u_1 = u'_1$, $w'_1 xu'yvzw'_2 \overset{*}{\Rightarrow} x_1 u'_1 y_1 v_1 z_1 = w$; and in case $v_1 = v'_1$, $w'_1 xuyv'zw'_2 \overset{*}{\Rightarrow} x_1 u_1 y_1 v'_1 z_1 = w$. This proves that $f(a)$ has the above property.

We now transform G_{fa} into a linear grammar G'_{fa}. First, we assume (using a standard construction) that all language names have nonempty domains. Second, we may assume that all productions of G_{fa} are of the form $A(\mathbf{x}) \to B_1(\phi_1(\mathbf{x}))$ $B_2(\phi_2(\mathbf{x})) \cdots B_k(\phi_k(\mathbf{x}))$ or of the form $A(\mathbf{x}) \to \psi(\mathbf{x})$, i.e., the ψ's between the B's have domain $\{\lambda\}$, see Lemma 2.3 of [21]. Next we change G_{fa} in such a way that for every nonterminal A of G_{fa} there exists $V_1 \subseteq V$ (note that V is the terminal alphabet of G_{fa}) such that, in a derivation of G_{fa}, $A(\mathbf{s})$ generates terminal strings w with $\text{alph}(w) = V_1$ only (where $\mathbf{s}$ is a sequence of actual arguments of A, i.e., $A(\mathbf{s})$ occurs in a sentential form of G_{fa}). To obtain this property, add "alph-information"

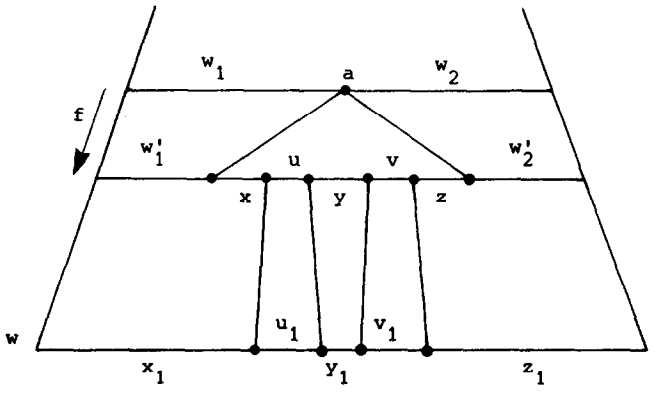

FIG. 2. The (P2)-like property.

to each nonterminal, using a construction similar to the one in the proof of Lemma 4.2 of [18] (with $h = \text{alph}$ and H is the set of subsets of V, in the terminology of that proof); see also the proof of Theorem 3.6 of [21]. Note that the productions of G_{fa} are still of the above form. We now turn G_{fa} into a K-extended linear basic macro grammar G'_{fa} with the following set P' of productions. If $A(\mathbf{x}) \to B_1(\phi_1(\mathbf{x}))\ B_2(\phi_2(\mathbf{x})) \cdots B_k(\phi_k(\mathbf{x}))$ is in P, then P' contains k productions $A(\mathbf{x}) \to w_1(\mathbf{x}) \cdots w_{i-1}(\mathbf{x})\ B_i(\phi_i(\mathbf{x}))\ w_{i+1}(\mathbf{x}) \cdots w_k(\mathbf{x})$, $1 \leqslant i \leqslant k$, where $w_j(\mathbf{x})$ is a fixed string over $V \cup X$ such that $B_j(\phi_j(\mathbf{x})) \overset{*}{\Rightarrow} w_j(\mathbf{x})$ in G_{fa} $(1 \leqslant j \leqslant k)$. If $A(\mathbf{x}) \to \psi(\mathbf{x})$ is in P, then it is also in P'. G'_{fa} is clearly a K-extended linear basic macro grammar, and $L(G'_{fa}) \subseteq L(G_{fa})$. To see that $f(a)$ can be changed into $L(G'_{fa})$ without changing $L(G)$, consider a derivation of G which uses $t \in f(a)$ in some derivation step. Suppose that in the derivation $S_{fa} \overset{*}{\Rightarrow} t$ of G_{fa} a derivation step is made by $A(\mathbf{s}) \Rightarrow B_1(\phi_1(\mathbf{s})) \cdots B_k(\phi_k(\mathbf{s}))$, and suppose that (for certain i and j) $B_i(\phi_i(\mathbf{s})) \overset{*}{\Rightarrow} u$ and $B_j(\phi_j(\mathbf{s})) \overset{*}{\Rightarrow} v$. Thus $t = xuyvz$ for certain $x, y, z \in V^*$, cf. Fig. 2. Now let $w_i(\mathbf{s}) \overset{*}{\Rightarrow} u'$ and $w_j(\mathbf{s}) \overset{*}{\Rightarrow} v'$ for some $u', v' \in V^*$ (note that the domains of the language names are nonempty and hence every element of $\mathbf{s}$ generates a terminal string). Then $B_i(\phi_i(\mathbf{s})) \overset{*}{\Rightarrow} w_i(\mathbf{s}) \overset{*}{\Rightarrow} u'$ and $B_j(\phi_j(\mathbf{s})) \overset{*}{\Rightarrow} w_j(\mathbf{s}) \overset{*}{\Rightarrow} v'$. Hence $xu'yvz$, $xuyv'z$, $xu'yv'z \in f(a)$, and (by the first change of G_{fa}) $\text{alph}(u) = \text{alph}(u')$ and $\text{alph}(v) = \text{alph}(v')$. From our discussion of the (P2)-like property of $f(a)$, it now follows that in our derivation step of G_{fa} we may replace $B_i(\phi_i(\mathbf{s}))$ by $w_i(\mathbf{s})$ or $B_j(\phi_j(\mathbf{s}))$ by $w_j(\mathbf{s})$. Repeating this argument shows that we may actually use a production of G'_{fa} in this derivation step. In this fashion we obtain a derivation of some t' in G'_{fa} such that in the given derivation of G we may use t' rather than t in the given derivation step (and, in fact, for *one* occurrence of a). Repeating this argument shows that we can use $L(G'_{fa})$ instead of $L(G_{fa})$ without changing $L(G)$. Hence, doing this for all $f(a)$, G can be turned into an $LB(K)$-extended iteration grammar, and the lemma is proved. ∎

As an immediate corollary we obtain the next theorem.

6.5. THEOREM. *Let K be full semi-*AFL, *and let L be a language with property* (P2). *If $L \in B^*(K)$, then $L \in H(K)$.*

Proof. If $L \in B^*(K)$, then $L \in H(B^n(K))$ for some n. Repeated application of Lemma 6.4 then shows that $L \in H(K)$. ∎

6.6. COROLLARY. *Let K be a full semi-*AFL *and L a language. If $\{w \# w^R \mid w \in L\} \in B^*(K)$, then $\{w \# w^R \mid w \in L\} \in H(K)$.*

Proof. $\{w \# w^R \mid w \in L\}$ has property (P2). ∎

We will now show that D_{CF} and $B^*(\text{FIN})$ are incomparable. Let $L \in \text{CF} - \text{ED0L}$ (see [12], or Corollary 3.2.18 of [16]), and consider $L_1 = \{w \# w^R \mid w \in L\}$: the language used in [13] to show that ETOL $\subsetneq$ INDEX. Clearly $L_1 \in 2DGSM(\text{CF})$, and, by Lemma 7.1 of [20], $L_1 \in \text{INDEX}$. Hence $L_1 \in D_{CF}$. However, $L_1 \in B^*(\text{FIN})$ would imply that $L_1 \in H(\text{FIN}) = \text{ETOL}$ by Corollary 6.6, and so L_1

and L are in EDTOL (see [34]). Thus $L_1 \in D_{CF} - B^*(\mathrm{FIN})$. Let L_2 be a generator of $H(B(\mathrm{FIN}))$; note that $H(B(\mathrm{FIN}))$ is a substitution-closed full principal AFL, cf. Corollary 4.6. If $L_2 \in D_{CF}$, then $H(B(\mathrm{FIN})) \subseteq 2DGSM(\mathrm{CF})$ and hence, by Proposition 6.2, $H(B(\mathrm{FIN})) \subseteq \mathrm{CF}$. Consequently $L_2 \in B^*(\mathrm{FIN}) - D_{CF}$.

Since D_{CF} and $B^*(\mathrm{FIN})$ are incomparable full semi-AFLs, $\hat{F}(D_{CF} \cup B^*(\mathrm{FIN}))$ is not substitution closed (Theorem 4.1 of [26]). Hence, by Theorem 5.5, $B^*(\hat{F}(D_{CF} \cup B^*(\mathrm{FIN}))) = B^*(D_{CF})$ is the union of an infinite hierarchy of full hyper-AFLs containing $B^*(\mathrm{FIN})$. This was one of the results we wanted to show in this section. Next we do the same for $B^*(D_{CF})$ and $B^*(D_{EB})$.

It suffices to show that D_{EB} and $B^*(D_{CF})$ are incomparable (as above this implies that $B^*(D_{EB})$ is the union of an infinite hierarchy of full hyper-AFLs containing $B^*(D_{CF})$). The language L_2 above is in $B^*(D_{CF})$ but not in D_{EB} (otherwise $H(B(\mathrm{FIN})) \subseteq B(\mathrm{FIN})$). Let $L_3 = \{w \# w^{\mathrm{R}} \mid w \in \mathrm{cut}(\mathrm{CUT})\}$. By Lemma 4.2, $\mathrm{cut}(\mathrm{CUT}) \in B(\mathrm{ONE})$. Hence $L_3 \in 2DGSM(B(\mathrm{FIN}))$, and, by Lemma 7.1 of [20], $L_3 \in \mathrm{INDEX}$. Thus $L_3 \in D_{EB}$. Assume that $L_3 \in B^*(D_{CF})$. By Corollary 6.6, $L_3 \in H(D_{CF})$ and hence $\mathrm{cut}(\mathrm{CUT}) \in H(D_{CF})$. Consequently, by Theorem 4.4, $\mathrm{CUT} \in E(D_{CF})$. Hence, by Corollary 3.5, $\mathrm{CUT} \in D_{CF}$, i.e., $\mathrm{CUT} \in 2DGSM(\mathrm{CF})$. By Corollary 5.6 of [16] (where $2DGSM(\mathrm{CF})$ is denoted $DCS(\mathrm{CF})$), $2DGSM(\mathrm{CF})$ is contained in the family $yT(\mathrm{REC})$ of top-down tree transformation languages. Hence $\mathrm{CUT} \in yT(\mathrm{REC})$, which contradicts Corollary 3.3. Consequently $L_3 \in D_{EB} - B^*(D_{CF})$.

We summarize the above results (and some of the previous sections) in the last theorem of this paper.

6.7. THEOREM. $B^*(\mathrm{FIN}) \subsetneq B^*(D_{CF}) \subsetneq B^*(D_{EB}) \subseteq N(\mathrm{INDEX}) \subsetneq \mathrm{INDEX}$. *The smallest full basic-*AFL $B^*(\mathrm{FIN})$ *is the union of an infinite hierarchy of full hyper-*AFL*s containing* $B(\mathrm{FIN})$. *The full basic-*AFL $B^*(D_{CF})$ *is the union of an infinite hierarchy of full hyper-*AFL*s containing* $B^*(\mathrm{FIN})$. *The full basic-*AFL $B^*(D_{EB})$ *is the union of an infinite hierarchy of full hyper-*AFL*s containing* $B^*(D_{CF})$. $N(\mathrm{INDEX})$ *is the largest full basic-*AFL, *properly contained in the full principal basic-*AFL INDEX.

The exact identity of $N(\mathrm{INDEX})$ remains an open problems (just as for $N(\mathrm{CF})$).

We finally note that we now have four essentially different languages in $\mathrm{INDEX} - \mathrm{ETOL}$. The first (and "lowest") is $\mathrm{CUT} \in B(\mathrm{ONE}) - \mathrm{ETOL}$. The second is $L_1 = \{w \# w^{\mathrm{R}} \mid w \in L\}$, where $L \in \mathrm{CF} - \mathrm{EDTOL}$ (e.g., L is the Dyck language): the first example of a language in $\mathrm{INDEX} - \mathrm{ETOL}$ (given in [13]). By the above, $L_1 \notin \hat{B}(\mathrm{CUT}) \subseteq B^*(\mathrm{FIN})$, i.e., L_1 is not reachable from CUT by full basic-AFL operations. The third is $L_3 = \{w \# w^{\mathrm{R}} \mid w \in \mathrm{cut}(\mathrm{CUT})\}$. Again, $L_3 \notin \hat{B}(L_1) \subseteq B^*(D_{CF})$, hence L_3 is not reachable from L_1 by full basic-AFL operations. Finally, the fourth is any generator L_4 of the full principal AFL INDEX. By the above, $L_4 \notin \hat{B}(L_3) \subseteq B^*(D_{EB}) \subseteq N(\mathrm{INDEX})$, and so, again, L_4 is not reachable from L_3 by full basic-AFL operations.

CONCLUSION

We have investigated the "AFL-structure" of the family of indexed languages, in particular in its higher regions, i.e., above $B(\text{FIN})$. We have established the existence of infinitely many full hyper-AFLs and of three full basic-AFLs properly contained in INDEX.

The following questions still have to be answered. Do there exist more full basic-AFLs inside INDEX? Is there a natural characterization of $N(\text{INDEX})$, e.g., by a restriction on the nested stack automaton? Is $N(\text{INDEX})$ full principal? Is there a natural operator O on families of languages such that $O(B(\text{FIN})) = \text{INDEX}$, but $O(\text{REG}) \subsetneq \text{INDEX}$? Are there any interesting full basic-AFLs, incomparable with INDEX, inside the context-sensitive languages?

ACKNOWLEDGMENTS

I wish to thank Peter Asveld for many stimulating conversations on AFL-theory. I am grateful to Giora Slutzki for many useful comments. This paper would not exist without the fascinating papers of Sheila Greibach.

REFERENCES

1. A. V. AHO, Indexed grammars—An extension of context-free grammars, *J. Assoc. Comput. Mach.* **15** (1968), 647–671.
2. A. V. AHO, Nested stack automata, *J. Assoc. Comput. Mach.* **16** (1969), 383–406.
3. A. V. AHO AND J. D. ULLMAN, A characterization of two-way deterministic classes of languages, *J. Comput. System Sci.* **4** (1970), 523–538.
4. P. R. J. ASVELD, Controlled iteration grammars and full hyper-AFLs, *Inform. and Control* **34** (1977), 248–269.
5. P. R. J. ASVELD, "Iterated Context-Independent Rewriting—An Algebraic Approach to Families of Languages," Doctoral dissertation, Twente University of Technology, 1978.
6. P. R. J. ASVELD, Space-bounded complexity classes and iterated deterministic substitution, *Inform. and Control* **44** (1980), 282–299.
7. P. R. J. ASVELD AND J. ENGELFRIET, Extended linear macro grammars, iteration grammars, and register programs, *Acta Inform.* **11** (1979), 259–285.
8. P. R. J. ASVELD AND J. VAN LEEUWEN, "Infinite Chains of Hyper-AFLs," TW-Memorandum 99, Twente University of Technology, 1975.
9. J. BERSTEL, "Transductions and Context-free Languages," Teubner Studienbücher Informatik, Teubner, Stuttgart, 1979.
10. P. A. CHRISTENSEN, Hyper-AFLs and ETOL systems, *in* "*L Systems*" (G. Rozenberg and A. Salomaa, Eds.), Lecture Notes in Computer Science Vol. 15, pp. 254–257, Springer-Verlag, Berlin, 1974.
11. P. J. DOWNEY, "Formal Languages and Recursion Schemes," Ph.D. Thesis, Report TR 16–74, Harvard University, 1974; "Proceedings Conf. on Biologically Motivated Automata Theory (IEEE), McLean, Va., 1974," pp. 54–58.
12. A. EHRENFEUCHT AND G. ROZENBERG, On some context-free languages that are not deterministic ETOL languages; *RAIRO Sér. Rouge* **11** (1977), 273–291.

13. A. EHRENFEUCHT, G. ROZENBERG, AND S. SKYUM, A relationship between ETOL languages and EDTOL languages, *Theoret. Comput. Sci.* **1** (1976), 325–330.
14. J. ENGELFRIET, Macro grammars, Lindenmayer systems and other copying devices, *in* "4th Int. Colloq. Automata Lang. and Programming" (A. Salomaa and M. Steinby, Eds.), Lecture Notes in Computer Science Vol. 52, pp. 221–229, Springer-Verlag, Berlin, 1977.
15. J. ENGELFRIET, Three hierarchies of transducers, *Math. Systems Theory* **15** (1982), 95–125.
16. J. ENGELFRIET, G. ROZENBERG, AND G. SLUTZKI, Tree transducers, *L* systems, and two-way machines, *J. Comput. System Sci.* **20** (1980), 150–202.
17. J. ENGELFRIET AND E. MEINECHE SCHMIDT, IO and OI, Parts I, II, *J. Comput. System Sci.* **15** (1977), 328–353; **16** (1978), 67–99.
18. J. ENGELFRIET, E. MEINECHE SCHMIDT, AND J. VAN LEEUWEN, Stack machines and classes of nonnested macro languages , *J. Assoc. Comput. Mach.* **27** (1980), 96–117.
19. J. ENGELFRIET AND S. SKYUM, Copying theorems, *Inform. Process. Lett.* **4** (1976), 157–161.
20. J. ENGELFRIET AND G. SLUTZKI, Bounded nesting in macro grammars, *Inform. and Control* **42** (1979), 157–193.
21. J. ENGELFRIET AND G. SLUTZKI, "Extended Macro Grammars and Stack Controlled Machines," TW-Memorandum No. 379, Twente University of Technology, January 1982; *J. Comput. System Sci* **29** (1984), 366–408.
22. M. J. FISCHER, "Grammars with Macro-like Productions," Ph.D. Thesis, Harvard University, 1968; 9th SWAT, pp. 131–142.
23. S. GINSBURG, "Algebraic and Automata-theoretic Properties of Formal Languages," North-Holland, Amsterdam, 1975.
24. S. GINSBURG AND E. H. SPANIER, On incomparable AFL, *J. Comput. System Sci.* **9** (1974), 88–108.
25. S. A. GREIBACH, Checking automata and one-way stack languages, *J. Comput. System Sci.* **3** (1969), 196–217.
26. S. A. GREIBACH, Chains of full AFL's; *Math. Systems Theory* **4** (1970), 231–242.
27. S. A. GREIBACH, Full AFLs and nested iterated substitution, *Inform. and Control* **16** (1970), 7–35.
28. S. A. GREIBACH, Syntactic operators on full semi-AFLs, *J. Comput. System Sci.* **6** (1972), 30–76.
29. S. A. GREIBACH, One way finite visit automata, *Theoret. Comput. Sci.* **6** (1978), 175–221.
30. S. A. GREIBACH, Hierarchy theorems for two-way finite state transducers, *Acta Inform.* **11** (1978), 89–101.
31. J. E. HOPCROFT AND J. D. ULLMAN, "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, Mass., 1979.
32. G. ROZENBERG AND A. SALOMAA, "The Mathematical Theory of L Systems," Academic Press, New York, 1980.
33. A. SALOMAA, "Marcos, Iterated Substitution and Lindenmayer AFLs," DAIMI PB-18, Aarhus University, 1973.
34. S. SKYUM, Decomposition theorems for various kinds of languages parallel in nature, *SIAM J. Comput.* **5** (1976), 284–296.
35. J. VAN LEEUWEN, *F*-iteration languages, memorandum, University of California, Berkeley, 1973.
36. J. VAN LEEUWEN, A generalization of Parikh's theorem in formal language theory, *in* "2nd Int. Colloq. Automata Lang. and Programming" (J. Loeckx, Ed.), Lecture Notes in Computer Science Vol. 14, pp. 17–26, Springer-Verlag, Berlin, 1974.
37. J. VAN LEEUWEN, A study of complexity in hyper-algebraic families, *in* "Automata, Languages, Development" (A. Lindenmayer, G. Rozenberg, Eds.), pp. 323–333, North-Holland, Amsterdam, 1976.
38. J. VAN LEEUWEN, Effective constructions in well-partially-ordered free monoids, *Discrete Math.* **21** (1978), 237–252.
39. D. WOOD, A note on Lindenmayer systems, Szilard languages, spectra, and equivalence, *Int. J. Comput. Inform. Sci.* **4** (1975), 53–62.