ON THE TIME AND TAPE COMPLEXITY OF HYPER(1)-AFL's [†]

Wilhelm J. Erni

Inst. f. Angew. Informatik
Universität Karlsruhe
D-7500 Karlsruhe
Kollegium am Schloß, Bau IV
Fed. Rep. Germany

Inst. f. Angew. Mathematik
Universität Heidelberg
D-6900 Heidelberg
Im Neuenheimer Feld 5
Fed. Rep. Germany

ABSTRACT. It is well known that the membership question for the smallest hyper-AFL is NP-complete. One may ask whether this is the case for the smallest hyper(1)-AFL, too. Thus we study the family of block-indexed languages. We show that this family is a hyper(1)-AFL which is not a hyper-AFL and that it is contained in the family of languages log(n)-tape reducible to the context-free languages. This implies that the family of block-indexed languages, together with the smallest hyper(1)-AFL, has a tractable membership question and tape complexity $\log^2(n)$. Finally we note that the set $\{ww \ / \ w \in \{a,b\}^*\}$ is not a block-indexed language.

1. INTRODUCTION. The notion of substitution has been the subject of many investigations in mathematics, for example in the algebraic theory of formal languages [8] and the mathematical theory of L-systems [10]. If L is a language over an alphabet V and τ is a substitution on V^*, then we define $\tau^{(\infty)}(L) = \bigcup_{n\geq 1} \tau^{(n)}(L)$ as usual [8] and speak of an iterated substitution. If $\alpha \in \tau(\alpha)$ for each $\alpha \in V$, then $\tau^{(\infty)}$ is called a nested iterated substitution. If P =

[†]This work constitutes part of a Ph. D. thesis and is supported by a German Graduate Fellowship.

$= \{\tau_i / i \in I\}$ is a finite set of substitutions over V with $I=\{1,\dots,m\}$ then we define $P^{(\infty)}(L) = \underbrace{\qquad}_{i_1 \dots i_k \in I^+} \tau_{i_1} \dots \tau_{i_k}(L)$ and speak of an iterated multiple substitution [5].

Now we are able to characterize several well known language families using the notion of substitution. For example the context-free languages are the smallest super-AFL, that is the smallest full AFL closed under nested iterated substitution [8] and the ETOL langua= ages are the smallest hyper-AFL, that is the smallest full AFL closed under iterated multiple substitution [10]. A notion between super-AFL and hyper-AFL is the notion of a hyper(1)-AFL, that is a full AFL closed under iterated substitution [8].

As mentioned above, nested iterated substitution can be characterized by context-free grammars. Similar iterated multiple substitution can be characterized by indexed grammars [11]. In fact the family of indexed languages is a hyper-AFL. We restrict the indexed grammars in a natural way to obtain so called block-indexed grammars which can be used to characterize iterated substitution and which can be described informally as follows.

It is easily seen that for each indexed grammar one can construct an equivalent one with only two indices, say f and g, and so that in the terminal derivations only indexed nonterminals of the form $Af^{i_1}g \dots f^{i_n}g$ are used, where $n>0$ and $i_k \geq 0$ for $1 \leq k \leq n$. Clearly, every index-block $f^j g$ can be interpreted as a counter with actual content j. If we allow at most m such index-blocks to be attached to a nonterminal and if we write those indexed nonterminals as $A[i_1,\dots,i_m]$ then the indexed grammars restricted in this way give an informal description of the m-block-indexed grammars. We say that L is a block-indexed language if there is an $m \geq 1$ and an m-block-indexed grammar which generates L.

The concept of block-indexed grammars turns out to be useful to prove several interesting properties concerning the generative capacity, and the time and tape complexity. In fact the family of block-indexed languages is shown to be a hyper(1)-AFL which does not contain, similar as the context-free languages, the language $\{ww/w \in \{a,b\}^*\}$. Furthermore each block-indexed language is accepted by a nondeterministic log(n)-tape bounded auxiliary pushdown automaton in polynomial time. This implies that the family of block-indexed languages is contained in the family of languages log(n)-tape reducible to the context-free languages [12]. Thus the family of block-indexed languages has polynomial time complexity, that is a tractable membership question, and tape complexity $\log^2(n)$, similar as the context-free languages. These properties indicate that the notion of a hyper(1)-AFL is more related to the notion of a super-AFL than to that of a hyper-AFL. In fact the language $\{ww/w \in \{a,b\}^*\}$ is an element of the smallest hyper-AFL and the membership question for the smallest hyper-AFL is NP-complete [1] , [9] .

Section 2 introduces the basic notions, where we define hyper(1)-AFL's and hyper-AFL's by means of iteration grammars [10] and give the necessary properties of auxiliary pushdown automata .

In section 3 we define the m-block-indexed grammars, $m \geq 1$, and present their basic properties. In this section Theorem 3.5 shows the main result which intuitively means that leftmost derivations in an m-block-indexed grammar G, $m \geq 1$, can be simulated by a nondeterministic log(n)-tape bounded auxiliary pushdown automaton M. The fact that we may assume that M accepts in polynomial time is guaranteed by Lemma 3.4 which intuitively says that a leftmost derivation according to G needs not to be too long in relation to the length of the generated word.

2. PRELIMINARIES. All terms used and not explicitely defined in the sequel may be found in [1,8,10,11].

Definition. Let $\mathcal{L}$ be a quasoid. An $\mathcal{L}$-iteration grammar is a 4-tuple $G=(V_N,V_T,S,P)$, where V_N is the alphabet of nonterminals, V_T is the alphabet of terminals with $V_N \cap V_T=\emptyset$, $S \in V_N$ is the start sym= bol, and $P=\{\tau_i / i \in I\}$ is a finite set of $\mathcal{L}$-substitutions defined on $V_N \cup V_T$ and with the property, that for each $i \in I=\{1,\dots,m\}$ and $\alpha \in V_N \cup V_T$, $\tau_i(\alpha) \in \mathcal{L}$ is a language over $V_N \cup V_T$. The language genera= ted by such a grammar is defined by

$$L(G)=\bigcup_{i_1\dots i_k \in I^+} \tau_{i_1} \dots \tau_{i_k} (S) \cap V_T^*$$

The family of languages generated by $\mathcal{L}$-iteration grammars is deno= ted by $\mathcal{L}_{\infty\text{-ITER}}$. By $\mathcal{L}_{m\text{-ITER}}$, we denote the subfamily of $\mathcal{L}_{\infty\text{-ITER}}$, generated by such grammars, where P consists of at most m elements, for some $m \geq 1$.

Example. Let $\mathcal{L}_{REG}$ be the family of regular languages and $G=(\{S,A,B\},\{a,b\},S,\{\tau_1,\tau_2,\tau_3\})$ be an $\mathcal{L}_{REG}$-iteration grammar, where

$\tau_1(S)=\{AB\}, \tau_1(A)=\{\varepsilon\}, \tau_1(B)=\{\varepsilon\}, \tau_1(a)=\{a\}, \tau_1(b)=\{b\}$

$\tau_2(S)=\{S\}\ , \tau_2(A)=\{aA\}, \tau_2(B)=\{aB\}, \tau_2(a)=\{a\}, \tau_2(b)=\{b\}$

$\tau_3(S)=\{S\}\ , \tau_3(A)=\{bA\}, \tau_3(B)=\{bB\}, \tau_3(a)=\{a\}, \tau_3(b)=\{b\}$

Then we have $L(G)=\{ww / w \in \{a,b\}^*\} \in (\mathcal{L}_{REG})_{\infty\text{-ITER}}$.

Theorem 2.1. (Asveld) $\mathcal{L}_{\infty\text{-ITER}}=\mathcal{L}_{2\text{-ITER}}$, for each quasoid $\mathcal{L}$.

Definition. Let $\mathcal{L}$ be a quasoid. $\mathcal{L}$ is called a hyper-AFL, if $\mathcal{L}=\mathcal{L}_{2\text{-ITER}}$, and $\mathcal{L}$ is called a hyper(1)-AFL, if $\mathcal{L}=\mathcal{L}_{1\text{-ITER}}$.

Example. Let $\mathcal{L}_1=\mathcal{L}_{REG}$ and for $i \geq 1$ let $\mathcal{L}_{i+1}=(\mathcal{L}_i)_{1\text{-ITER}}$. Then $(\mathcal{L}_1)_{\infty\text{-ITER}}$ is the smallest hyper-AFL and $\mathcal{L}_\infty=\bigcup_{i\geq 1} \mathcal{L}_i$ is the smallest hyper(1)-AFL [5,11].

Let $\mathcal{L}_{\mathrm{PTIME}}$ be the family of languages acceptable by determinis= tic polynomial time bounded Turing machines. S.A.Cook has characte= rized $\mathcal{L}_{\mathrm{PTIME}}$ by the nondeterministic log(n)-tape bounded auxiliary pushdown automata [4].

<u>Definition</u>. A nondeterministic log(n)-tape bounded auxiliary push-down automaton, short aux-PDA, is a nondeterministic log(n)-tape bounded Turing machine which has an additional unbounded pushdown store. Let $\mathcal{L}_{\mathrm{PPTIME}}$ be the family of languages acceptable in poly-nomial time by nondeterministic log(n)-tape bounded aux-PDA's.

<u>Theorem 2.2</u>. (Sudborough) $\mathcal{L}_{\mathrm{PPTIME}}=\mathrm{LOG}(\mathcal{L}_{\mathrm{CF}})$, the family of langu-ages log(n)-tape reducible to the family of context-free languages.

3. <u>THE FAMILY OF BLOCK-INDEXED LANGUAGES</u>

Here we will study $\mathcal{L}_{\infty\text{-IND}}$, the family of block-indexed languages [2], and prove $\mathcal{L}_{\infty} \subseteq \mathcal{L}_{\infty\text{-IND}} \subseteq \mathcal{L}_{\mathrm{PPTIME}}$, the main result of this pa-per.

<u>Definition</u>. Let $m \in \mathbb{N}$. An m-block-indexed grammar, short m-IND gram-mar, is a 4-tuple $G=(V_N,V_T,S[0,\dots,0],P)$ with the following proper-ties ($\mathbb{N}$ denotes the set of natural numbers and $\mathbb{N}_0 = \mathbb{N}\cup\{0\}$):

(1) V_N is the alphabet of nonterminals or variables

(2) V_T is the alphabet of terminals with $V_N \cap V_T = \emptyset$

(3) Let $\overline{V}_N=\{A[i_1,\dots,i_m]/A \in V_N,\ (i_1,\dots,i_m) \in \mathbb{N}_0^m\}$ be the set of m-block-indexed variables, short m-IND variables. Then $S[0,\dots,0] \in \overline{V}_N$ is the start variable.

(4) Let $\widetilde{\mathbb{N}} = \{0+,1+,2+,\dots\}$ be an infinite set of abstract symbols. Then $\widetilde{V}_N = \{A[j]/\ A \in V_N,\ j \in \mathbb{N}_0 \cup \widetilde{\mathbb{N}}\}$ is the set of metavariables. Now P is a finite subset of $\widetilde{V}_N \times \{1,2,\dots,m\} \times (\widetilde{V}_N \cup V_T)^*$. An element $(A[j],k,\omega) \in P$ is called a metaproduction and is written in the form $A[j] \xrightarrow[(k)]{} \omega$.

<u>Definition</u>. Let $G=(V_N,V_T,S[0,\dots,0],P)$ be an m-IND grammar. The indexfunction $\delta: (\mathbb{N}\cup\widetilde{\mathbb{N}}) \times \mathbb{N} \to \mathbb{N}$ is defined by

$$\delta(j\ ,\ i) = j \qquad , \quad \text{for } j \in \mathbb{N}$$
$$\delta(j+,\ i) = j + i\ , \quad \text{for } j+ \in \widetilde{\mathbb{N}}\ .$$

With a metaproduction $p \in P$ of the form $p=A_0[j_0] \xrightarrow[(k)]{} x_0A_1[j_1]x_1 \dots$ $\dots x_{t-1}A_t[j_t]x_t$, where $1\leq k\leq m$, $A_0,A_1,\dots,A_t \in V_N$, $j_0,j_1,\dots,j_t \in \mathbb{N}\cup\widetilde{\mathbb{N}}$ and $x_0,x_1,\dots,x_t \in V_T^*$, we associate a set

$$\bar{p} = \{A_0[0,\dots,0,\ \delta(j_0,i),\ i_{k+1},\dots,i_m] \longrightarrow$$
$$x_0A_1[0,\dots,0,\ \delta(j_1,i),\ i_{k+1},\dots,i_m]\ x_1\ \dots$$
$$\dots x_{t-1}A_t[0,\dots,0,\ \delta(j_t,i),\ i_{k+1},\dots,i_m]x_t/(i,i_{k+1},\dots,i_m)\in \mathbb{N}^{m-k+1}\}$$

of productions. $\bar{P} = \bigcup_{p\in P} \bar{p}$ is the set of productions of G.

<u>Definition</u>. Let $G=(V_N,V_T,S[0,\dots,0],P)$ be an m-IND grammar and $\chi \in (\bar{V}_N \cup V_T)^*\bar{V}_N(\bar{V}_N \cup V_T)^*$, $\Psi\in(\bar{V}_N \cup V_T)^*$. We write $\chi \Rightarrow \Psi$, if $\chi = \chi_1\ A[i_1,\dots,i_m]\chi_2$, $A[i_1,\dots,i_m]\to\omega\in\bar{P}$, $\Psi = \chi_1\omega\chi_2$ and speak of the application of the production $A[i_1,\dots,i_m]\to\omega$ in a derivation step. If in addition $\chi_1 \in V_T^*$, then we write $\chi \underset{lm}{\Longrightarrow} \Psi$ and speak of the application of the production $A[i_1,\dots,i_m]\to\omega$ in a leftmost derivation step. A sequence $\Psi_0, \Psi_1,\dots,\Psi_k$ such that $k\geq1$, $\chi = \Psi_0$, $\Psi=\Psi_k$ and $\Psi_0\Rightarrow\Psi_1 \Rightarrow \dots \Rightarrow\Psi_k$ ($\Psi_0 \underset{lm}{\Longrightarrow} \Psi_1 \underset{lm}{\Longrightarrow} \dots \underset{lm}{\Longrightarrow} \Psi_k$) is called a (leftmost) derivation of Ψ from χ with k steps. Furthermore we extend the relation $\Longrightarrow$ ($\underset{lm}{\Longrightarrow}$) to the relation $\overset{*}{\Longrightarrow}$ ($\underset{lm}{\overset{*}{\Longrightarrow}}$) as usual.

<u>Definition</u>. Let $G=(V_N,V_T,S[0,\dots,0],P)$ be an m-IND grammar and $\Longrightarrow$ ($\underset{lm}{\Longrightarrow}$) the relation according to G, as defined above. Then we call

$$L(G)=\{w\in V_T^* \ /\ S[0,\dots,0] \overset{*}{\Longrightarrow} w\}=\{w\in V_T^* \ /\ S[0,\dots,0] \underset{lm}{\overset{*}{\Longrightarrow}} w\}$$

the language generated by G. Furthermore $\mathcal{L}_{m\text{-IND}}=\{L(G)/G$ is an m-IND grammar$\}$ is called the family of m-block-indexed languages.
Finally, we call $\mathcal{L}_{\infty\text{-IND}}=\bigcup_{m\geq1}\mathcal{L}_{m\text{-IND}}$ the family of block-indexed languages.

<u>Example</u>. Consider a 2-IND grammar $G_1=(\{S,A\},\{a,a^E,b,b^E\},S,P_1)$, where $P_1=\{S[0]\xrightarrow[(1)]{}aA[1]b, A[0+]\xrightarrow[(1)]{}aA[1+]b, A[1+]\xrightarrow[(1)]{}a^EA[0+]b^E, A[0]\xrightarrow[(2)]{}\varepsilon\}$.

Then for $p=A[1+]\xrightarrow[(1)]{}a^EA[0+]b^E$ we have $\bar{p}=\{A[1+i,j]\longrightarrow a^EA[0+i,j]b^E \,/\, i,j\in\mathbb{N}\}$. The following leftmost derivation is possible:
$S[0,0]\xRightarrow[lm]{} aA[1,0]b \xRightarrow[lm]{} aa^EA[0,0]b^Eb \xRightarrow[lm]{} aa^Eb^Eb$.
Finally $L_1=L(G_1)=\{wf(w)/w\in D_1\}\in\mathcal{L}_{\infty\text{-IND}}$, where $D_1\subseteq\{a,a^E\}^*$ is the Dyck set on $\{a\}$ and $f(w)=(h(w))^R$, $w\in\{a,a^E\}^*$, with a homomorphism $h:\{a,a^E\}^*\to\{b,b^E\}^*$, $h(a)=b$, $h(a^E)=b^E$.

<u>Lemma 3.1</u>. (Albert) Let $m\in\mathbb{N}$. Then $\{ww/w\in\{a,b\}^*\}\notin\mathcal{L}_{m\text{-IND}}$ is a super-AFL with $\mathcal{L}_{CF}\subsetneq\mathcal{L}_{1\text{-IND}}\subsetneq\cdots\subsetneq\mathcal{L}_{m\text{-IND}}\subsetneq\mathcal{L}_{(m+1)\text{-IND}}\subsetneq\cdots\subsetneq\mathcal{L}_{\infty\text{-IND}}\subsetneq\mathcal{L}_{IND}$, the family of indexed languages. If $L_1\in\mathcal{L}_{m\text{-IND}}$ with $L_1\subseteq V_1^*$ for an alphabet V_1, and for each $a\in V_1$ we have $L_a\in\mathcal{L}_{m\text{-IND}}$ with $L_a\subseteq V_a^*$ for an alphabet V_a, then for the substitution τ defined by $\tau(a)=L_a$, for each $a\in V_1$, it is true, that the iterated substitution $\tau^{(\infty)}$ has the property $\tau^{(\infty)}(L_1)\in\mathcal{L}_{(m+1)\text{-IND}}$.

<u>Theorem 3.2</u>. $\mathcal{L}_{\infty\text{-IND}}$ is a hyper(1)-AFL which is not a hyper-AFL.

<u>Proof</u>. $\mathcal{L}_{\infty\text{-IND}}\subseteq(\mathcal{L}_{\infty\text{-IND}})_{1\text{-ITER}}$ is true by definition. For the proof of $(\mathcal{L}_{\infty\text{-IND}})_{1\text{-ITER}}\subseteq\mathcal{L}_{\infty\text{-IND}}$ consider an $L\in(\mathcal{L}_{\infty\text{-IND}})_{1\text{-ITER}}$. By definition $L=L(G)$ for an $\mathcal{L}_{\infty\text{-IND}}$-iteration grammar $G=(V_N,V_T,S,\{\tau\})$. Because $\tau(\alpha)\in\mathcal{L}_{\infty\text{-IND}}=\underbrace{\bigcup}_{m\geq 1}\mathcal{L}_{m\text{-IND}}$, for each $\alpha\in V_N\cup V_T$, there exists $m\in\mathbb{N}$, such that $\tau(\alpha)\in\mathcal{L}_{m\text{-IND}}$, for each $\alpha\in V_N\cup V_T$. Now again by definition we have for the iterated substitution $\tau^{(\infty)}$ that $L(G)=\tau^{(\infty)}(\{S\})\cap V_T^*$ and by Lemma 3.1 $L=L(G)\in\mathcal{L}_{(m+1)\text{-IND}}\subseteq\mathcal{L}_{\infty\text{-IND}}$ follows. Thus we have proved $\mathcal{L}_{\infty\text{-IND}}=(\mathcal{L}_{\infty\text{-IND}})_{1\text{-ITER}}$ and $\mathcal{L}_{\infty\text{-IND}}$ is a hyper(1)-AFL by definition. That $\mathcal{L}_{\infty\text{-IND}}$ is not a hyper-AFL follows from the fact, that $\{ww/w\in\{a,b\}^*\}\notin\mathcal{L}_{\infty\text{-IND}}$ is, as shown in the Preliminaries, an element of the smallest hyper-AFL $(\mathcal{L}_1)_{\infty\text{-ITER}}$.

J.Engelfriet has shown [5], that $\{wcwcw/w\in\{a,b\}^*\}\notin\mathcal{L}_\infty$. By the previous theorem we can improve to $\{ww/w\in\{a,b\}^*\}\notin\mathcal{L}_\infty$.

Similar to the case of context-free grammars, a simple normalform (see below) for m-IND grammars can be given. This will facilitate the proof of $\mathcal{L}_{m\text{-IND}}\subseteq\mathcal{L}_{PPTIME}$.

Definition. An m-IND grammar $G=(V_N,V_T,S[0,..,0],P)$ is called in normalform, if P contains only metaproductions of the following form: (1.1) $A[0+]\xrightarrow[(k)]{}B[1+]$ (1.2) $A[1+]\xrightarrow[(k)]{}B[0+]$

(1.3) $A[0]\xrightarrow[(k)]{}B[0]$ (2) $A[0+]\xrightarrow[(1)]{}B[0+]\,C[0+]$

(3) $A[0]\xrightarrow[(m)]{}a$ (4) $S[0]\xrightarrow[(m)]{}\varepsilon$

where $A\in V_N$, B, $C\in V_N-\{S\}$, $a\in V_T$ and $k\in\{1,...,m\}$.

Lemma 3.3. (Albert) For each m-IND grammar $G'=(V_N',V_T',S'[0,...,0],P')$ there exists an m-IND grammar $G=(V_N,V_T,S[0,...,0],P)$ in normalform, such that $V_T'=V_T$ and $L(G')=$ $=L(G)$. Furthermore for each m-IND grammar $G=(V_N,V_T,S[0,...,0],P)$ in normalform there is a constant $c_0=c_0(G)\in\mathbb{N}$, such that the following holds true: If $w\in L(G)$, then there is a leftmost derivation $S[0,...,0]=\Psi_0\xRightarrow[lm]{}\Psi_1\xRightarrow[lm]{}\;...\;\xRightarrow[lm]{}\Psi_k=w$, where each m-IND variable $A[i_1,...,i_m]\in\overline{V}_N$ occuring in one of the $\Psi_0,\Psi_1,...,\Psi_k\in V_T^*\,\overline{V}_N^*$ has the property $\{i_1,...,i_m\}\subseteq\{1,2,...,c_0\cdot|w|\}$.

Lemma 3.4. If $\varepsilon\notin L=L(G)\in\mathcal{L}_{m\text{-IND}}$, where $G=(V_N,V_T,S[0,..,0],P)$ is a m-IND grammar in normalform, then there exists a constant $c_1=c_1(G)\in\mathbb{N}$ such that for each $w\in L(G)$ we have a leftmost derivation

$$S[0,...,0]=\varphi_0\xRightarrow[lm]{}\varphi_1\xRightarrow[lm]{}\;...\;\xRightarrow[lm]{}\varphi_t=w$$

with $t\le c_1\cdot|w|^{m+2}$ and where each m-IND variable $A[i_1,...,i_m]\in\overline{V}_N$ occuring in one of the $\varphi_0,\varphi_1,.....,\varphi_t\in V_T^*\,\overline{V}_N^*$ has the property

$\{i_1,\ldots,i_m\} \subseteq \{1,2,\ldots,c_1\cdot|w|\}$.

<u>Proof</u>. (Sketch) The proof is based on the following observations: Let $G=(V_N,V_T,S[0,..,0],P)$ be the given m-IND grammar in normalform. By Lemma 3.3 there exists a constant $c_0=c_0(G)\in\mathbb{N}$, such that for each $w\in L(G)$, $|w|=n$ there exists a leftmost derivation.

$$(*)\qquad S[0,..,0]=\Psi_0 \xrightarrow[lm]{} \Psi_1 \xrightarrow[lm]{} \cdots \xrightarrow[lm]{} \Psi_k = w ,$$

where each m-IND variable $A[i_1,..,i_m]\in\overline{V}_N$ occuring in one of the $\Psi_0, \Psi_1,\ldots, \Psi_k \in V_T^*\overline{V}_N^*$ has the property

$$(**)\quad \{i_1,..,i_m\}\subseteq\{1,2,..,c_0\cdot n\} .$$

Because $G=(V_N,V_T,S[0,..,0],P)$ is in normalform, in $(*)$ exactly n-1 expanding productions, that is productions associated with metaproductions of the form $A[0+] \xrightarrow[(1)]{} B[0+]\ C[0+]$, are used. This fact is indicated by the following "decomposition" of $(*)$:

$$S[0,..,0]=\Psi_{r_0+1} \overset{*}{\underset{lm}{\Longrightarrow}} \Psi_{r_1} \xrightarrow[lm]{} \Psi_{r_1+1} \overset{*}{\underset{lm}{\Longrightarrow}} \Psi_{r_2} \xrightarrow[lm]{} \Psi_{r_2+1} \overset{*}{\underset{lm}{\Longrightarrow}} ..$$

$$\ldots \overset{*}{\underset{lm}{\Longrightarrow}} \Psi_{r_j} \xrightarrow[lm]{} \Psi_{r_j+1} \overset{*}{\underset{lm}{\Longrightarrow}} .. \overset{*}{\underset{lm}{\Longrightarrow}} \Psi_{r_{n-1}} \xrightarrow[lm]{} \Psi_{r_{n-1}+1} \overset{*}{\underset{lm}{\Longrightarrow}} \Psi_{r_n} =$$

$=\Psi_k$ where $r_0=-1$, $0\le r_1\le\ldots\le r_n=k$ and expanding productions are used only in the derivation steps $\Psi_{r_j} \xrightarrow[lm]{} \Psi_{r_j+1}$, $j=1,..,n-1$. In all other "intermediate" derivations $\Psi_{r_j+1} \overset{*}{\underset{lm}{\Longrightarrow}} \Psi_{r_{j+1}}$, $j=0,1,..,n-1$, only productions associated with metaproductions of the form $A[0+]\xrightarrow[(i)]{} B[1+]$, $A[1+]\xrightarrow[(i)]{} B[0+]$, $A[0]\xrightarrow[(i)]{} B[0]$, $A[0]\xrightarrow[(m)]{} a$ where $A\in V_N$, $B,\ C\in V_N-\{S\}$, $a\in V_T$ and $i\in\{1,..,m\}$, are applied. The suitable "shrinkage" of these derivations, using property $(**)$, is the intuitive idea to find the desired leftmost derivation

$$S[0,..,0]=\Psi_0=\varphi_0=\Psi_{s_0} \xrightarrow[lm]{} \varphi_1=\Psi_{s_1} \xrightarrow[lm]{} \cdots \xrightarrow[lm]{} \varphi_t=\Psi_{s_t}=\Psi_k=w$$

where $t\le c_1\cdot|w|^{m+2}$, $\{r_1,..,r_n\}\subseteq\{s_0,s_1,..,s_t\}\subseteq\{0,1,..,k\}$ and each m-IND variable $A[i_1,..,i_m]\in\overline{V}_N$ occuring in one of the $\varphi_0, \varphi_1,..$ $..,\varphi_t\in V_T^*\overline{V}_N^*$ has the property $\{i_1,..,i_m\}\subseteq\{1,2,..,c_1\cdot n\}$. The details about this "shrinkage" may be found in [6].

Theorem 3.5. $\mathcal{L}_{\infty} \subseteq \mathcal{L}_{\infty\text{-IND}} \subseteq \mathcal{L}_{\text{PPTIME}}$

Proof. By Theorem 3.2 it suffices to prove $\mathcal{L}_{\infty\text{-IND}} \subseteq \mathcal{L}_{\text{PPTIME}}$ and by the definition of $\mathcal{L}_{\infty\text{-IND}} = \bigcup_{m\geq 1} \mathcal{L}_{m\text{-IND}}$ the proof is complete, if we can show $\mathcal{L}_{m\text{-IND}} \subseteq \mathcal{L}_{\text{PPTIME}}$, for each $m \in \mathbb{N}$. We consider only the case m=2. Then the reader will be able to see how the general case $m \in \mathbb{N}$ can be shown.

Thus let $L \in \mathcal{L}_{2\text{-IND}}$. Then by Lemma 3.1 and 3.3 $L - \{\varepsilon\} = L(G)$, for some 2-IND grammar $G = (V_N, V_T, S[0,0], P)$ in normalform with $S[0] \xrightarrow[(2)]{} \varepsilon \notin P$. Note that $\varepsilon \notin L(G) \in \mathcal{L}_{\text{PPTIME}}$ implies $L = L(G) \cup \{\varepsilon\} \subseteq \mathcal{L}_{\text{PPTIME}}$, because $\{\varepsilon\} \in \mathcal{L}_{\text{PPTIME}}$ is closed under union [7]. We design a nondeterministic log(n)-tape bounded aux-PDA M which accepts L(G) .

Given as input word $w=\varepsilon$, M reaches a dead state, that is a state from which we could never reach an accepting state. Given an input word w, $|w|=n\geq 1$, M intuitively traces out (nondeterministically) each leftmost derivation according to G of the form:

(✱) $S[0,0] = \varphi_0 \xrightarrow[lm]{} \varphi_1 \xrightarrow[lm]{} \cdots \xrightarrow[lm]{} \varphi_r \xrightarrow[lm]{} \varphi_{r+1} \xrightarrow[lm]{} \cdots \xrightarrow[lm]{} \varphi_t$

with $r<t \in \mathbb{N}$ and each 2-IND variable $A[i,j] \in \overline{V}_N$ occuring in one of the $\varphi_0, \varphi_1, \ldots, \varphi_t \in V_T^* \overline{V}_N^*$ has the property $\{i,j\} \subseteq \{1,2,\ldots,c_1 \cdot n\}$, where $c_1 = c_1(G) \in \mathbb{N}$ is the constant of Lemma 3.4 corresponding to G. To simulate a leftmost derivation of the form (✱) M stores $X_r \in \overline{V}_n^*$, the nonterminal part of the current sentential form $\varphi_r = u_r X_r \in V_T^* \overline{V}_N^*$, $u_r \in V_T^*$, on its pushdown tape $\mathcal{P}_0$, where each $A[i,j] \in \overline{V}_N$ is encoded as $Af^i g f^j g$, $f,g \notin V_N \cup V_T$. M's input head guarantees that u_r is consistent with the input w, that is u_r is an initial subword of w. To simulate a derivation step $\varphi_r \xrightarrow[lm]{} \varphi_{r+1}$, such that only 2-IND variables $Af^i g f^j g$ with $\{i,j\} \subseteq \{1,2,\ldots,c_1 \cdot n\}$ appear on $\mathcal{P}_0$ (during the simulation), M uses five log(n)-tape bounded auxiliary tapes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_5$, where elements from $\{1,2,\ldots,c_1 \cdot n\}$ are stored in binary notation. Suppose, M has reached the end of the simulation process for (✱). Then M may decide to enter an "end-check phase",

examining whether $\varphi_t = w$ or $\varphi_t \neq w$.

Let us make more precise how M simulates derivation step $\varphi_r \xRightarrow[\text{lm}]{} \varphi_{r+1}$. Because $G=(V_N, V_T, S[0,0], P)$ is in normalform with $S[0] \xrightarrow[(2)]{} \varepsilon \notin P$, each production applied in (*) is associated with a metaproduction of the form (1.1) $A[0+] \xrightarrow[(k)]{} B[1+]$ (1.2) $A[1+] \xrightarrow[(k)]{} B[0+]$ (1.3) $A[0] \xrightarrow[(k)]{} B[0]$ (2) $A[0+] \xrightarrow[(1)]{} B[0+]C[0+]$ (3) $A[0] \xrightarrow[(2)]{} a$ where $A \in V_N$, $B,C \in V_N - \{S\}$, $a \in V_T$ and $k \in \{1,2\}$. We assume, that each metaproduction of G is "embedded" in M's finite control.

At the start of M's computation with the input word w, where $w=a_1 \ldots a_n$ is of length n, $\mathcal{P}_0$ is empty, $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_5$ consist of blanks only, the finite control is in the initial state and the input head scans the blank to the left of a_1. First M stores $c_1 \cdot n$ in $\mathcal{A}_5$. This may be realized by a standard technique for log(n)-tape bounded aux-PDA's. That is, M's input head makes a sweep from left (the blank to the left of a_1) to right (the blank to the right of a_n), such that for each symbol $a \in \{a_1, \ldots, a_n\}$ scanned by the input head, M adds c_1 to $\mathcal{A}_5$. After this, the input head moves back over a_1. Then M pushes Sgg, the encoding of $S[0,0]$, in the top region of $\mathcal{P}_0$.

Suppose, M has simulated of (*) the segment

$$S[0,0] = \varphi_0 \xRightarrow[\text{lm}]{} \varphi_1 \xRightarrow[\text{lm}]{} \cdots \xRightarrow[\text{lm}]{} \varphi_r,$$

where $r < t$, $\varphi_r = u_r X_r$, $u_r \in V_T^*$,
$X_r = A_r[i_r, j_r] X'_r \in \overline{V}_N^+$ with $A_r[i_r, j_r] \in \overline{V}_N$, $\{i_r, j_r\} \subseteq \{1,2,\ldots,c_1 \cdot n\}$, and $X'_r \in \overline{V}_N^*$. That is, we assume that $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ consist of blanks only, $\mathcal{A}_5$ contains $c_1 \cdot n$, that the input head has scanned of w the initial subword u_r and that X_r is on $\mathcal{P}_0$, in the encoding mentioned above. In other words, in the top region of $\mathcal{P}_0$ we have $Af^{i_r}gf^{j_r}g$, where $A = A_r$.

Now M nondeterministically chooses by its finite control a metaproduction with A on the left side (if no such metaproduction exists, we reach the dead state). Then M simulates the derivation step $\varphi_r \xrightarrow[\text{lm}]{} \varphi_{r+1}$ in (*), according to the type of metaproduction, with which the production to be applied in this step is associated, as follows:

<u>type (1.1)</u>: $A[0+] \xrightarrow[(k)]{} B[1+]$, $k \in \{1,2\}$

For k=2 we replace $Af^{i_r}gf^{j_r}g$ by $Bf^{i_r}gf^{j_r+1}g$ and check $\{i_r, j_r+1\} \subseteq \{1,2,\dots,c_1 \cdot n\}$. To do this M pops $Af^{i_r}gf^{j_r}g$ from the top region of $\mathcal{P}_0$ and stores i_r in $\mathcal{A}_1$ and j_r in $\mathcal{A}_2$. Then M adds 1 to $\mathcal{A}_2$ and checks whether the content of $\mathcal{A}_2$ is not greater than the content of $\mathcal{A}_5$. If this is not the case, M reaches the dead state. Otherwise M pushes $Bf^{i_r}gf^{j_r+1}g$ in the top region of $\mathcal{P}_0$, using the contents of $\mathcal{A}_1$ and $\mathcal{A}_2$, which at the end of this simulation phase consist of blanks only. Type (1.1) for k=1 and the types (1.2),(1.3) are omitted, but should be clear from the considered types.

<u>type (2)</u>: $A[0+] \xrightarrow[(1)]{} B[0+]\ C[0+]$.

We have to replace the word $Af^{i_r}gf^{j_r}g$ by $Bf^{i_r}gf^{j_r}gCf^{i_r}gf^{j_r}g$. To do this M pops $Af^{i_r}gf^{j_r}g$ from the top region of $\mathcal{P}_0$ and stores i_r in $\mathcal{A}_1, \mathcal{A}_2$ and j_r in $\mathcal{A}_3, \mathcal{A}_4$. Then M pushes $Cf^{i_r}gf^{j_r}g$ in the top region of $\mathcal{P}_0$, using the contents of $\mathcal{A}_2, \mathcal{A}_4$. After this M pushes $Bf^{i_r}gf^{j_r}g$ in the top region of $\mathcal{P}_0$, using the contents of $\mathcal{A}_1, \mathcal{A}_3$. At the end of this simulation phase $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ consist of blanks only.

<u>type(3)</u>: $A[0] \xrightarrow[(2)]{} a$

We have to check, whether Agg is in the top region of $\mathcal{P}_0$ and "consistent" with the symbol scanned by the input head. Thus first A is popped from $\mathcal{P}_0$. Then, if a g is on the top of $\mathcal{P}_0$, it is popped, and if again a g is on the top of $\mathcal{P}_0$ and the symbol scanned by the input head is a, then the last g is popped, too and the input head moves one cell right on the input tape; otherwise we reach the dead state.

Suppose, M has reached the "end" of the simulation process for a derivation of the form (*), $S[0,0] \xrightarrow[lm]{*} \varphi_t$, for some $t>0$. Now M may decide to enter the "end-check phase", examining, whether $\varphi_t=w$. This phase is realized by M's finite control, the input head and $\mathcal{P}_0$. If $\mathcal{P}_0$ is empty and the input head scans the blank (right of a_n), M accepts. Otherwise we reach the dead state.

To prove the Theorem we still have to show, that M accepts in polynomial time. It is easily seen, that by construction the number of computation steps needed by M - given as input word w, $|w|=n\geq 1$ - for the simulation phase of an initial sweep to store $c_1 \cdot n$ in $\mathcal{A}_5$ and of the derivation step according to type (1) - (3) is at most $c_2 \cdot n^2$, with some constant $c_2=c_2(G)\in \mathbb{N}$, depending only on G and not on n. Thus by Lemma 3.4 for each input word $w \in L(G)$, $|w|=n\geq 1$, M has an accepting sequence of computation steps with a length s, where

$$s \leq (1+c_1 \cdot n^4)\cdot c_2 \cdot n^2 \leq c_3 \cdot n^6 \quad , \; c_3 = 2\cdot c_1(G)\cdot c_2(G) \; .$$

Problem. Let $L_1=L(G_1)=\{wf(w)/w \in D_1\}$, where G_1 is the 2-IND grammar as defined above. Can you prove that $L_1 \in \mathcal{L}_{\infty\text{-IND}} - \mathcal{L}_{\infty}$ or even that $L_1 \in \mathcal{L}_{\infty\text{-IND}} - (\mathcal{L}_1)_{\infty\text{-ITER}}$?

Acknowledgement. I wish to thank J.Albert for many useful discussions.

REFERENCES.

1 Aho,A.V., Hopcroft,J., Ullman,J., The Design and Analysis of Computer Algorithms, Addison Wesley Publishing Company, Massachusetts, 1974.

2 Albert,J., Über indizierte und m-Block-indizierte Grammatiken, Dissertation, Universität Karlsruhe, 1976.

3 Asveld,P.R.J., Rational, Algebraic and Hyper-Algebraic Extensions of Families of Languages, Memorandum Nr. 90, Technische

Hoogeschool Twente, Onderafdeling der Toegepaste Wiskunde, 1976.

4 Cook,S.A., Characterization of Pushdown Machines in Terms of Time-Bounded Computers, Journal of the Association for Computing Machinery, 18 (1971) 4-18.

5 Engelfriet,J., Iterating Iterated Substitution, Memorandum Nr. 143, Technische Hoogeschool Twente, Onderafdeling der Toegepaste Wiskunde, 1976.

6 Erni, W., Thesis to appear at the Institut für angewandte Informatik und formale Beschreibungsverfahren, Universität Karlsruhe, 1977.

7 Erni ,W., Some Further Languages Log-Tape Reducible to the Context-Free Languages, Research Report of the Institut für angewandte Informatik und formale Beschreibungsverfahren, Universität Karlsruhe, 1976.

8 Ginsburg,S., Algebraic and Automata-Theoretic Properties of Formal Languages, North-Holland Publishing Company, Amsterdan, 1975.

9 Leeuwen,J.van, The Membership Question for ETOL-Languages is Polynomially Complete, Information Processing Letters 3 (1975) 138-143.

10 Rozenberg,G., Salomaa,A., L-Systems, Lecture Notes in Computer Science 15 (1974).

11 Salomaa,A., Formal Languages, Academic Press, New York, 1973.

12 Sudborough,I.H., On Languages Log-Tape Reducible to Context-Free Languages, Proceedings of 1976 Conference on Information Sciences and Systems at Johns Hopkins University.