stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 108/79 APRIL
(DEPARTMENT OF COMPUTER SCIENCE)

P.R.J. ASVELD

SPACE-BOUNDED COMPLEXITY CLASSES AND ITERATED DETERMINISTIC SUBSTITUTION

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68A30, 68A20

ACM-Computing Reviews-categories: 5.23, 5.25

Space-Bounded Complexity Classes and Iterated Deterministic Substitution *)

by

Peter R.J. Asveld

ABSTRACT

We investigate the effect on the space complexity when a language family K is extended by means of iterated λ-free deterministic substitution to the family $\eta(K)$. If each language in K is accepted by a one-way nondeterministic multi-tape Turing machine within space S(n) for some monotonic space bound $S(n) \geq \log n$, then $\eta(K)$ is included in NSPACE(S(n)). Thus for each monotonic space bound $S(n) \geq n$, the inclusion $K \subseteq$ NSPACE(S(n)) implies that $\eta(K)$ is also included in NSPACE(S(n)). An implication similar to the latter one also holds for DSPACE(S(n)).

Consequently, some well-known space-bounded complexity classes such as the families of (non)deterministic context-sensitive languages, of two-way (non)deterministic nonerasing stack automaton languages, and *PSPACE* are AFL's closed under intersection and iterated λ-free deterministic substitution. On the other hand no space-bounded complexity class which includes DPSACE(log n) is closed under controlled iterated λ-free (non)deterministic substitution.

KEY WORDS & PHRASES: *space-bounded complexity classes, (controlled) iterated parallel rewriting, iterated (deterministic) substitution, closure properties.*

*) This report will be submitted for publication elsewhere.

1. INTRODUCTION

The operation of homomorphism is one of the central notions in studying closure properties of language families. It has been generalized succesfully to the concept of (language-theoretic) substitution (cf. Ginsburg, 1975, Ginsburg et al. 1969, Hopcroft and Ullman 1969 and the references mentioned there). When we apply a substitution τ on a word w, we choose for each occurrence of a symbol α in w a word from a given language $\tau(\alpha)$ and we replace that occurrence of α by that choosen word from $\tau(\alpha)$.

In the study of certain classes of parallel rewriting systems, viz. the so-called deterministic Lindenmayer systems (Herman and Rozenberg, 1975), the notion of *deterministic substitution* has been studied as another generalization of the homomorphism concept (Asveld and Engelfriet, 1977). Deterministic substitution differs from ordinary or *nondeterministic substitution* by the way we replace (occurrences of) symbols in words: if we apply a deterministic substitution τ on a word w, then we have to choose for each symbol α a word v_α from the language $\tau(\alpha)$ and we have to replace each occurrence of α in w by v_α, i.e. by the same word from $\tau(\alpha)$.

If for a substitution τ, each language $\tau(\alpha)$ is a singleton language (i.e. $\tau(\alpha)$ contains exactly one word), then both notions of substitution coincide with the concept of homomorphism. But in general these notions are independent, i.e. there exist language families closed under deterministic but not under nondeterministic substitution, and vice versa (Asveld and Engelfriet, 1977).

Deterministic substituion has originally been introduced in the analysis of certain complexity problems (Rounds, 1973). However it was first studied more systematically in establishing the relation between iterating finite sets of deterministic substitutions and, on the other hand, generalized deterministic Lindenmayer systems (the so-called deterministic K-iteration grammars, Asveld and Engelfriet, 1977) and certain classes of polyadic program schemes (Asveld and Engelfriet, 1979). The relation between iterating finite sets of nondeterministic substitutions and generalized nondeterministic Lindenmayer systems (the nondeterministic K-iteration grammars) was known previously; Van Leeuwen, 1973; Salomaa, 1973; Asveld, 1977.

The smallest family containing the regular sets, which is closed under iterated deterministic substitution is EDTOL, i.e. the family of extended deterministic tabled context-independent Lindenmayer or EDTOL languages (Rozenberg, 1973; Asveld and Engelfriet, 1977). Each EDTOL language may be obtained by iterating a finite set of homomorphisms over an alphabet V, starting from an initial symbol $S \in V$, and finally intersecting with Σ^* for some terminal alphabet $\Sigma \subseteq V$. Replacing these finite set of homomorphisms by a finite set of [non]deterministic K-substitutions yields the rather abstract notion of [non]deterministic K-iteration grammar (A substitution τ is a K-substitution, if $\tau(\alpha) \in K$ for each α, where K is a given language family); cf. Van Leeuwen, 1973; Salomaa, 1973; Asveld, 1977, 1978; Asveld and Engelfriet 1977, 1979. So the family $\eta(K)$ [$H(K)$] of all languages generated by [non]deterministic K-iteration grammars obviously depends on K, and it includes EDTOL whenever K contains all singleton languages.

A further generalization leads to the notion of Γ-controlled K-iteration grammar, in which not all sequences of substitutions are allowed in the iteration process, but only a restricted set of sequences, prescribed by a control language, is applied. This control language is a language over the finite set of substitutions and it is taken from a given family Γ of control languages (Asveld, 1977, 1978; Asveld and Engelfriet 1977). The family of languages generated by Γ-controlled [non]deterministic K-iteration grammars is denoted by $\eta(\Gamma,K)$ [$H(\Gamma,K)$].

Operators on language families like η and H are usually called extensions, since under weak assumptions on K and Γ they are indeed extensive, i.e. $K \subseteq \eta(K)$ and $\Gamma,K \subseteq \eta(\Gamma,K)$, and similarly for H. Although extensions are a promising tool in recent more algebraic approaches to the theory of grammars and languages (Van Leeuwen, 1973, 1976; Asveld, 1977, 1978; Asveld and Engelfriet, 1977, 1979), in this paper we restrict our attention to the operators η and H only.

Let now X denote either η or H. Since in any case X is extensive, the following two problems arise quite naturally (cf. Van Leeuwen, 1976):

(A) Given the complexity of K and Γ, e.g. $K,\Gamma \subseteq C$ for some complexity class C. Can we bound the complexity of languages in the extended family $X(K)$, or $X(\Gamma,K)$ respectively? Or, equivalently, does there exist a complexity class C' for which $X(K) \subseteq C'$, or $X(\Gamma,K) \subseteq C'$ respectively, holds?

(B) Are there complexity classes C_0 which are fixed points of such extension operators, i.e. which satisfy $X(C_0) = C_0$ or $X(C_0,C_0) = C_0$?

Note that if we are able to solve (A) with $C' = C$, we also have a solution for (B), viz. $C_0 = C = C'$.

Van Leeuwen (1976) considered these problems in case that $X = H$ and the complexity classes are space-bounded classes. He proved that if $K \subseteq$ DSPACE(S(n)), then $H(K) \subseteq$ DSPACE(S(n)) provided that $S(n) \geq n \log n$; i.e. for $S(n) \geq n \log n$, DSPACE(S(n)) and similarly NSPACE(S(n)) are fixed points of H.

In this paper we study the instance of these problems in which $X = \eta$ and the complexity classes are space-bounded.

Let K be a family containing each finite alphabet, and let Γ be a family closed under intersection with regular sets and arbitrary finite substitution. We show that if K and Γ are included in 1-NSPACE(S(n)) —— i.e. each language in both K and Γ is accepted by a one-way nondeterministic multi-tape Turing machine within space S(n) —— for some monotonic space bound $S(n) \geq \log n$, then $\eta(\Gamma,K)$ and $\eta(K)$ are included in NSPACE(S(n)). (This result can be considered as a generalization of the inclusion EDTOL $\subseteq$ NSPACE(log n) established by Jones and Skyum (1977), since for the family REG of regular languages we have η(REG,REG) = η(REG) = EDTOL, cf. Asveld and Engelfriet, 1977). Consequently, for $S(n) \geq n$ the inclusion $K \subseteq$ NSPACE(S(n)) implies that $\eta(K)$ is included in NSPACE(S(n)) as well. A similar conclusion also holds for DSPACE(S(n)) with $S(n) \geq n$. Thus for each $S(n) \geq n$ with $S(2n) \leq c.S(n)$ for some constant c, both DSPACE(S(n)) and NSPACE(S(n)) are AFL's closed under intersection and iterated λ-free deterministic substitution. (These families are however not closed under controlled iterated λ-free deterministic substitution). In particular this applies to the families of (non)deterministic context-sensitive languages, (non)deterministic two-way nonerasing stack automaton languages, and to *PSPACE*.

Apart from the present introductory section this paper contains three other sections. Section 2 consists of some preliminaries concerning complexity classes, iterated (non)deterministic substitution, and controlled iteration grammars. We conclude Section 2 with an important lemma on the length of derivations according to (controlled) λ-free iteration grammars. In section 3 we establish our main result, viz. we solve problem (A) with $X = \eta$, C = 1-NSPACE(S(n)), and C' = NSPACE(S(n)) for $S(n) \geq \log n$. The proof consists of a generalization of the main ideas developed by Jones and Skyum (1977) in showing that the family EDTOL is included in NSPACE(log n); cf. also Erni, 1977; Harju, 1977; Sudborough, 1977. This result implies another solution of problem (A) for $X = \eta$, viz. $C = C'$ = NSPACE(S(n)) with $S(n) \geq n$. By slightly changing Van Leeuwen's (1976) algorithm we obtain a similar solution with respect to deterministic space-bounded complexity classes, i.e. $C = C'$ = DSPACE(S(n)) with $S(n) \geq n$. In Section 4 we directly derive from our main result a solution of problem (B), viz. $\eta(C_0) = C_0$ if C_0 equals NSPACE(S(n)) or DSPACE(S(n)) for $S(n) \geq n$, and we establish the above mentioned closure properties for these families. With respect to problem (B) we also show that for each complexity class C which includes DSPACE(log n) we have neither $\eta(C,C) = C$ nor $H(C,C) = C$.

2. PRELIMINARIES

We assume the reader to be familiar with the rudiments of formal languages, space- and time-bounded Turing machines, and Lindenmayer systems (cf. e.g. Hopcroft and Ullman, 1975; Herman and Rozenberg, 1975).

An *AFL* or *Abstract Family of Languages* (Ginsburg, 1975; Ginsburg et al. 1969) is any nontrivial family closed under union, concatenation, Kleen +, λ-free homomorphism, inverse homomorphism and intersection with regular languages.

Let S(n) be a monotonic function, i.e. $m \leq n$ implies $S(m) \leq S(n)$. Let DSPACE(S(n)) [NSPACE(S(n))] be the family of languages accepted by [non]-deterministic two-way multi-tape Turing machines which scan no more than S(n) tape cells on any storage tape during a computation on an input of length n. Similarly, DTIME(T(n)) [NTIME(T(n))] denotes the family of languages accepted by [non]deterministic two-way multi-tape Turing machines which operate within time T(n) on all inputs of length n. The family of languages accepted by nondeterministic one-way multi-tape Turing machines within space-bound S(n), is denoted by 1-NSPACE(S(n)). The families *PSPACE*,

$\mathcal{P}$ and $\mathcal{NP}$ are defined as usual, viz. by

$$\mathcal{PSPACE} = \bigcup_{k\geq 1} \mathrm{DSPACE}(n^k) = \bigcup_{k\geq 1} \mathrm{NSPACE}(n^k),$$

$$\mathcal{P} = \bigcup_{k\geq 1} \mathrm{DTIME}(n^k), \quad \mathcal{NP} = \bigcup_{k\geq 1} \mathrm{NTIME}(n^k).$$

We now turn to operations on languages, in particular to substitution and its variants. We first recall the formal definition of nondeterministic and deterministic substitution.

DEFINITION 2.1. Let K be a family of languages. A *nondeterministic K-substitution* or *nK-substitution* on an alphabet V is a mapping $\tau: V \to K$. This mapping is extended to words over V by $\tau(\lambda) = \{\lambda\}$, $\tau(\alpha_1\ldots\alpha_n) = \tau(\alpha_1)\ldots\tau(\alpha_n)$, $\alpha_i \in V$ $(1 \leq i \leq n)$, and to languages L over V by $\tau(L) = \cup\{\tau(w) \mid w \in L\}$.

A *deterministic K-substitution* or *dK-substitution* on an alphabet V is also a mapping $\tau: V \to K$. But now τ is extended to words w over V by $\tau(w) = \{h(w) \mid h$ is a homomorphism such that $h(\alpha) \in \tau(\alpha)$ for each $\alpha \in V\}$. The extension of τ to languages $L \subseteq V^*$ is as usual: $\tau(L) = \cup\{\tau(w) \mid w \in L\}$.

If $\tau(\alpha) \subseteq V^*$ for each α in V, then the [non]deterministic K-substitution $\tau: V \to K$ is called a *[non]deterministic K-substitution over* V.

A family K is closed under iterated [non]deterministic substitution if for each language L in K and for each finite set U of [non]deterministic K-substitutions over some alphabet, the language $U^*(L)$ —— defined by $U^*(L) = \cup\{\tau_p(\ldots(\tau_1(L))\ldots) \mid p \geq 0;\ \tau_i \in U,\ 1 \leq i \leq p\}$ —— is in K. □

EXAMPLE. Let τ be a regular substitution over $\{a,b\}$ defined by $\tau(a) = a^*$ and $\tau(b) = \{b^j \mid 1 \leq j \leq n\}$. If we interpret τ as a deterministic substitution, then $\tau(abba) = \{a^ib^jb^ja^i \mid i \geq 0;\ 1 \leq j \leq n\}$. Considering τ as a nondeterministic substitution yields however $\tau(abba) = \{a^ib^jb^ka^\ell \mid i,\ell \geq 0;\ 1 \leq j,k \leq n\}$. □

Although (controlled) iteration grammars have originally been introduced in order to generalize certain parts of Lindenmayer system theory, they form an appropriate tool to study iterated substitution. In the following definition we collect the main notions quoted from (Van Leeuwen, 1973; Salomaa, 1973; Asveld, 1977, 1978; Asveld and Engelfriet, 1977, 1979).

DEFINITION 2.2. Let Γ and K be families of languages. A *[non]deterministic K-iteration* or *dK-iteration [nK-iteration] grammar* $G = (V,\Sigma,U,S)$ consists of an alphabet V, a terminal alphabet $\Sigma \subseteq V$, an initial symbol $S \in V$, and a finite set U of [non]deterministic K-substitutions over V. The language L(G) generated by G is defined by $L(G) = U^*(S) \cap \Sigma^*$. A *$\Gamma$-controlled dK-iteration [nK-iteration] grammar* $G = (V,\Sigma,U,M,S)$ consists of a dK-iteration [nK-iteration] grammar (V,Σ,U,S) provided with a *control language* $M \subseteq U^*$ satisfying $M \in \Gamma$, whereas the language L(G) generated by G is defined by $L(G) = M(S) \cap \Sigma^*$, where $M(S) = \cup\{\tau_p(\ldots(\tau_1(S))\ldots) \mid \tau_1\ldots\tau_p \in M\}$. The family of languages generated by dK-iteration [nK-iteration] grammars is denoted by $\eta(K)$ [$H(K)$]. Similarly, $\eta(\Gamma,K)$ [$H(\Gamma,K)$] denotes the family of languages generated by Γ-controlled dK-iteration [nK-iteration] grammars. □

EXAMPLE. Consider the regularly controlled K-iteration grammar $G = (V,\Sigma,U,M,S)$ where $\Sigma = \Delta \cup \{\$\}$, $V = \Sigma \cup \{S\}$, $U = \{\sigma,\tau\}$, $M = \{\sigma\tau^{2n} \mid n \geq 0\}$, and σ and τ are defined by

$$\begin{array}{lll} \sigma(S) = L_0 & & L_0 \subseteq \Delta^* \text{ and } L \in K \ (S,\$ \notin \Delta) \\ \sigma(\alpha) = \{\alpha\} & & \text{if } \alpha \in V - \{S\} \\ \tau(S) = \{S\$S\} & & \\ \tau(\alpha) = \{\alpha\} & & \text{if } \alpha \in V - \{S\}. \end{array}$$

If U consists of dK-substitutions, then $L(G) = \{(w\$)^{2^{2n}-1}w \mid n \geq 0;\ w \in L_0\}$ whereas in case σ and τ are nK-substitutions the following holds: $L(G) = \cup\{(L_0\$)^{2^{2n}-1}L_0 \mid n \geq 0\}$. □

From Definitions 2.1 and 2.2 it follows immediately that a family K, containing {S} for each symbol S, which is closed under intersection with Σ^* for each alphabet Σ, is closed under iterated [non]deterministic substitution if and only if $\eta(K) \subseteq K$ [$H(K) \subseteq K$ respectively]. Similarly, we call each family K satisfying $\eta(K,K) \subseteq K$ [$H(K,K) \subseteq K$] and the above mentioned simple conditions, closed under *controlled iterated [non]deterministic substitution*. For each family K containing U^* for each finite alphabet U, closure of K under controlled iterated [non]deterministic substitution implies that K is also closed under (uncontrolled) iterated [non]deterministic substitution.

For each word x, let $|x|$ denote the length of x.

In the following lemma we show that each word x derivable by a λ-free (controlled) iteration grammar can be obtained by a derivation of length at most linear in $|x|$, provided Γ and K satisfy some simple conditions.

LEMMA 2.3. *Let* K *be a family containing all finite alphabets.*
(1) *Let* Γ *be a family closed under finite substitution and intersection with regular sets, and let* $G = (V,\Sigma,U,M,S)$ *be a* λ*-free* Γ*-controlled* dK-*iteration* [nK-*iteration*] *grammar. Then there exists a* λ*-free* Γ*-controlled* dK-*iteration* [nK-*iteration*] *grammar* $H = (V_H,\Sigma,U_H,M_H,S)$ *such that* $L(H) = L(G)$*, and for each* x *in* L(H) *there is a control word* $\tau_1\ldots\tau_k$ *in* M_H *such that* $x \in \tau_k\ldots\tau_1(S)$ *and* $k \leq 2.|x|$.
(2) *For each* λ*-free* dK-*iteration* [nK-*iteration*] *grammar* $G = (V,\Sigma,U,S)$ *there exists a* λ*-free* dK-*iteration* [nK-*iteration*] *grammar* $H = (V_H,\Sigma,U_H,S)$ *such that* $L(H) = L(G)$*, and for each* x *in* L(H) *there is a string* $\tau_1\ldots\tau_k$ *over* U *such that* $x \in \tau_k\ldots\tau_1(S)$ *and* $k \leq 2.|x|$.

Proof. (1) Since G is λ-free each step in a derivation $w_2 \in \tau(w_1)$ for some τ in U, is either splitting, i.e. $|w_1| < |w_2|$, or stationary, i.e. $|w_1| = |w_2|$. In a possible derivation of a word x there are at most $|x| - 1$ splitting steps, but in general there is no bound on the number of stationary rewriting steps. As certain length preserving subderivations may correspond to the identity transformation they could be repeated any number of times (viz. $w \in u(w)$ for some $u \in U^+$, implies that $w \in u^n(w)$ for each $n \geq 1$) without increasing the length of the intermediate string. In this way long control words may give rise to derivations of relatively short strings.

The main idea is that we extend the control language M in the following way. For each control word m in M which gives rise to a derivation containing sequences $s_1,\ldots,s_\ell$ of consecutively stationary steps we add a finite number of new control words to M. These new control words are obtained from m as follows: for each subset $\{t_1,\ldots,t_n\}$ of $\{s_1,\ldots,s_\ell\}$ we replace in m the substring corresponding with t_i $(1 \leq i \leq n)$ by a new equivalent length-preserving substitution. (Since there is only a finite number of length-preserving substitutions over a fixed alphabet, U_H is finite). This construction implies that for each word x in L(H) there is a derivation of x according to H such that, whenever a stationary step occurs in that derivation this step is immediately followed by a splitting step. Therefore each word x in L(H) possesses a derivation according to H of length at most $2.|x|$.

We now turn to the formal construction of H.

Since Γ is closed under finite substitution and intersection with regular sets, it is closed under a-NGSM mappings, i.e. mappings induced by nondeterministic generalized sequential machines with accepting states (An a-NGSM is a nondeterministic finite automaton which can output a finite number of symbols for each input symbol; cf. e.g. (Hopcroft and Ullman, 1969) for formal definitions). Let $V = \{\alpha_1,\ldots,\alpha_k\}$, $V_H = V\cup\{F\}$ where F is a symbol not in V, and define U_H by $U_H = \{\tau' | \tau \in U\} \cup \{[\tau,q] | \tau \in U,\ q \in Q\}$ where the finite set Q is defined by $Q = \{<X_1,\ldots,X_k> | X_i \subseteq V,\ 1 \le i \le k\}$.

Consider the a-NGSM $T = (Q,U,U_H,\delta,q_0,Q_F)$ where Q is the set of states, $q_0 = <\{\alpha_1\},\ldots,\{\alpha_k\}>$ is the initial state, $Q_F = \{q_0\}$ is the set of final states, U is the input alphabet, U_H is the output alphabet, and δ is a mapping from $Q \times U$ to the finite subsets of $Q \times U_H^*$ defined by

$$\delta(<X_1,\ldots,X_k>,\tau) = \{(q_0,\tau') | <X_1,\ldots,X_k> = q_0\} \cup$$

$$\{(<\tau(X_1) \cap V,\ldots,\tau(X_k) \cap V>, \lambda),\ (q_0,[\tau,<X_1,\ldots,X_k>])\}.$$

The new control language M_H is defined by $M_H = T(M)$.

We briefly describe the effect of the new control language M_H. For each substitution τ in U occurring in a control word from M the a-NGSM T guesses nondeterministically one of the following three possibilities (cf. the definition of δ):

<u>Case 1</u>: The derivation step according to τ is splitting.

This corresponds to the transition $(q_0,\tau') \in \delta(q_0,\tau)$ of T where τ' is defined by

$$\tau'(\alpha_i) = \tau(\alpha_i) \qquad \text{for each } i\ (1 \le 1 \le k)$$

$$\tau'(F) = \{F\}.$$

Thus τ is replaced by τ' and T remains in state q_0.

Case 2: The derivation step according to τ is stationary and the next step will also be stationary.

We erase the occurrence of τ in the control word while we keep track of the stationary effect of τ by means of a change of state $<X_1,\ldots,X_k>$ into $<\tau(X_1)\cap V,\ldots,\tau(X_k)\cap V>$.

Case 3: The derivation step according to τ is stationary but the next step will be splitting.

In this case τ is the last substitution in a sequence of consecutively stationary steps. The ultimate effect of this sequence (including the stationary behaviour of τ) is performed by a new substitution $[\tau,<X_1,\ldots,X_k>]$ by which τ is replaced, while T returns to state q_0. This new substitution is defined by

$$[\tau,<X_1,\ldots,X_k>](F) = \{F\}$$

$$[\tau,<X_1,\ldots,X_k>](\alpha_i) = \tau(X_i)\cap V \qquad \text{if } \tau(X_i)\cap V \neq \emptyset$$

$$[\tau,<X_1,\ldots,X_k>](\alpha_i) = \{F\} \qquad \text{if } \tau(X_i)\cap V = \emptyset$$

The definitions of M_H and τ' imply that $L(G) \subseteq L(H)$; the formal proofs that $L(H) \subseteq L(G)$ and that H has the desired properties are left to the reader.

(2) Let $V_H = V\cup\{F\}$, where F is new. Define for each $u \in U^+$ a substitution σ_u over V_H by

$$\sigma_u(\alpha) = u(\alpha)\cap V \qquad \text{if } u(\alpha)\cap V \neq \emptyset$$

$$\sigma_u(\alpha) = \{F\} \qquad \text{otherwise.}$$

We call σ_u and σ_v $(u,v \in U^+)$ equivalent if $\sigma_u(\alpha) = \sigma_v(\alpha)$ for each α in V_H. Let $[\sigma_u]$ denote the corresponding equivalence class of σ_u. Then U_H is defined by $U_H = \{\tau' \mid \tau \in U\} \cup \{[\sigma_u] \mid u \in U^+\}$, where τ' is as above. Since there is only a finite number of length-preserving substitutions over V_H, the set $\{[\sigma_u] \mid u \in U^+\}$ is finite. □

Note that Lemma 2.3(1) may be viewed as a generalization to controlled iteration grammars of a similar result (viz. Theorem 3.4 in Asveld and Van

Leeuwen, 1975) concerning controlled ETOL systems.

3. THE MAIN RESULT

In this section we show that (under the assumptions of Lemma 2.3(1)) for each Γ-controlled λ-free dK-iteration grammar G the membership problem $x \in L(G)$? is solvable in NSPACE(S(n)) provided that $S(n) \geq \log n$ and membership in both Γ and K can be determined in 1-NSPACE(S(n)). First, we slightly generalize a few concepts and an auxiliary result concerning EDTOL systems originating from (Jones and Skyum, 1977; cf. Erni, 1977, Harju, 1977; Sudborough, 1977) to dK-iteration grammars (Definition 3.1 and Lemma 3.2).

Let $G_0 = (V,\Sigma,U,S)$ be a λ-free dK-iteration grammar with $V = \{\alpha_1,\ldots,\alpha_k\}$ and $S = \alpha_1$. Let x be a word over Σ of length n, and let \$ be a symbol not in V.

<u>DEFINITION 3.1.</u> A *configuration* (with respect to $x \in \Sigma^*$ and V) is a k-tuple $\bar{x} = \langle x_1,\ldots,x_k\rangle$ where each x_i is either \$ or a nonempty subword of x. A configuration $\bar{x}$ is called *consistent* if there exists a sequence of dK-substitutions $u \in U^*$ such that for each i $(1 \leq i \leq k)$ either $x_i = \$$ or $x_i \in u(\alpha_i)$.

The derivation relations $\overset{\tau}{\longmapsto}$ $(\tau \in U)$ between configurations $\bar{x} = \langle x_1,\ldots,x_k\rangle$ and $\bar{y} = \langle y_1,\ldots,y_k\rangle$ are defined as follows: $\bar{x} \overset{\tau}{\longmapsto} \bar{y}$ if for all i $(1 \leq i \leq k)$, if $y_i \neq \$$ then $y_i = h_{\bar{x}}(w)$ with $w \in \tau(\alpha_i)$ and the partial homomorphism $h_{\bar{x}}: V \to \Sigma^*$ is defined by $h_{\bar{x}}(\alpha_j) = x_j$ iff $x_j \in \Sigma^*$ $(1 \leq j \leq k)$. □

For each $w \in V^*$, alph(w) is the subalphabet of V consisting of those symbols which occur in w.

<u>LEMMA 3.2.</u>

(1) *Let* $\bar{x}$ *and* $\bar{y}$ *be configurations. If* $\bar{x}$ *is consistent and* $\bar{x} \overset{\tau}{\longmapsto} \bar{y}$ *for some* $\tau \in U$, *then* $\bar{y}$ *is consistent.*

(2) *Let* $v,w \in V^*$ *be such that* $x \in u(v)$ *and* $v \in \tau(w)$ *for some* $\tau \in U$, $u \in U^*$ *and let* $\bar{x}$ *be a consistent configuration such that for each* i, $x_i \in u(\alpha_i)$ *if* $\alpha_i \in \text{alph}(v)$ *and* $x_i = \$$ *otherwise. Then there exists a consistent configuration* $\bar{x}'$ *such that* $\bar{x} \overset{\tau}{\longmapsto} \bar{x}'$ *and for each* i, $x'_i \in u\tau(\alpha_i)$ *if*

$\alpha_i \in \mathrm{alph}(w)$ *and* $x'_i = \$$ *otherwise.*

(3) $x \in L(G_0)$ *if and only if there exists a sequence of configurations* $\bar{x}^{(0)} \overset{\tau_1}{\vdash} \bar{x}^{(1)} \overset{\tau_2}{\vdash} \ldots \overset{\tau_q}{\vdash} \bar{x}^{(q)}$ *such that* $\bar{x}^{(0)} = \langle \xi_1, \ldots, \xi_k \rangle$ *where* $\xi_i = \alpha_i$ *if* $\alpha_i \in \mathrm{alph}(x)$ *and* $\xi_i = \$$ *otherwise, and* $\bar{x}^{(q)} = \langle x, \$, \ldots, \$ \rangle$.

Proof. (1) Let $u \in U^*$, such that for each i $(1 \le i \le k)$ either $x_i = \$$ or $x_i \in u(\alpha_i)$. If $y_i \in \Sigma^*$ (i.e. $y_i \neq \$$) then there exists a word $w \in \tau(\alpha_i)$ and $y_i = h_{\bar{x}}(w)$, where the partial homomorphism $h_{\bar{x}}: V \to \Sigma^*$ is defined by $h_{\bar{x}}(\alpha_j) = x_j$ iff $x_j \in \Sigma^*$ $(1 \le j \le k)$. Then for each i either $y_i = \$$ or $y_i \in u\tau(\alpha_i)$, i.e. $\bar{y}$ is consistent.

(2) Let for each i, z_i be an element in $\tau(\alpha_i)$. Then we define x'_i by

$$x'_i = h_{\bar{x}}(z_i) \qquad \text{if } \alpha_i \in \mathrm{alph}(w)$$

$$x'_i = \$ \qquad \text{if } \alpha_i \notin \mathrm{alph}(w)$$

where the partial homomorphism $h_{\bar{x}}$ is defined as under (1). Since $\alpha_i \in \mathrm{alph}(w)$ implies $\mathrm{alph}(z_i) \subseteq \mathrm{alph}(v)$, we have $x'_i = h_{\bar{x}}(z_i) \in \Sigma^*$ whenever $\alpha_i \in \mathrm{alph}(w)$. Therefore $x'_i \in u\tau(\alpha_i)$ if $\alpha_i \in \mathrm{alph}(w)$ and $x'_i = \$$ if $\alpha_i \notin \mathrm{alph}(w)$. Thus $\bar{x}'$ is consistent and $\bar{x} \overset{\tau}{\vdash} \bar{x}'$.

(3) This follows from (1), (2) and a straightforward inductive argument. □

The interested reader is invited to compare Lemma 3.2(3) with the Equivalence Theorem in (Asveld and Engelfriet, 1979) and similarly Lemma 3 in (Jones and Skyum, 1977) with Theorem 2.9 in (Downey, 1974).

We are now ready to prove the main theorem. Let $G = (V,\Sigma,U,M,S)$ be a Γ-controlled λ-free dK-iteration grammar. In order to determine whether a given word $x \in \Sigma^*$ belongs to $L(G)$, it is sufficient (1) to decide whether $x \in L(G_0)$ where $G_0 = (V,\Sigma,U,S)$ is the uncontrolled λ-free dK-iteration grammar corresponding to G, (2) to keep track of the sequence $\tau_1 \ldots \tau_q$ of dK-substitutions used in a possible derivation of x according to G_0, and (3) to determine whether $\tau_1 \ldots \tau_q$ is in M.

In the algorithm to be presented below the implementation of (2) and (3) is straightforward. In implementing (1) we use simulation of a derivation by means of a sequence of configurations instead of storing complete sentential forms according to G_0 or G (cf. Lemma 3.2(3)).

<u>THEOREM 3.3</u>. *Let* $S(n) \geq \log n$, *and let* Γ *and* K *be families satisfying the assumptions of Lemma* 2.3.

(1) *If both* Γ *and* K *are included in* 1-NSPACE(S(n)), *then* $\eta(\Gamma,K) \subseteq$ NSPACE(S(n)).

(2) *If* $K \subseteq$ 1-NSPACE(S(n)), *then* $\eta(K) \subseteq$ NSPACE(S(n)).

Proof. In view of Lemma 3.2(3) and the remarks above it will be clear that the algorithm in Figure 3.1 determines whether $x \in L(G)$ for some given Γ-controlled λ-free dK-iteration grammar $G = (V,\Sigma,U,M,S)$ and an arbitrary string $x \in \Sigma^*$. In this algorithm we use the following auxiliary concepts and notational conventions:

- SUB(x) denotes the set of all nonempty substrings of a given string x.
- \ is the left quotient operator, i.e. $\backslash: \Sigma^* \times \Sigma^* \to \Sigma^*$ is a partial function defined by $u\backslash w = v$ iff $uv = w$. We write $u\backslash w\downarrow$ whenever $u\backslash w$ is defined.
- We assume that for each τ in U and each α in V, there has been given a nondeterministic one-way algorithm $A(\tau,\alpha)$ accepting the language $\tau(\alpha)$ within space S(n). Similarly, let CONTROL be a nondeterministic one-way algorithm accepting the control language M within space S(n). We write $A \leftarrow \alpha$ when the one-way algorithm A reads the next symbol of its input string (or rather when the next input symbol α is given to A and A continues its computation), and we write $A\downarrow$ when the algorithm A halts in an accepting state.

We will now analyse the amount of space needed in executing the algorithm of Figure 3.1.

Firstly, by Lemma 2.3(1) x has a derivation according to G or to G_0 of length at most $2.|x|$. So the corresponding control word m satisfies $|m| \leq 2n$, and hence the question whether m is in M can be resolved within $\mathcal{O}(S(n))$ space by the algorithm CONTROL.

The second problem deals with storing the configurations. By definition each entry x_i in a configuration $\bar{x}$ is either equal to \$ or to a substring of the input x. In the latter case x_i is completely determined by its starting and its ending position in the string x, i.e. by two natural numbers in between 1 and n which can be stored using $\mathcal{O}(\log n)$ bits. Thus storing the complete configuration $\bar{x}$ requires $\mathcal{O}(\log n)$ space.

```
1   for $i$ := 1 to $k$ do
2       $x_i$ := if $\alpha_i \in alph(x)$ then $\alpha_i$ else \$ fi; od;
3   $\bar{x}$ := $<x_1,\ldots,x_k>$;
4   while $\bar{x} \neq <x,\$,\ldots,\$>$ do
5       guess $\tau \in U$;
6       $i$ := 1;
7       while $i \leq k$ do
8           guess $y_i \in \{\$\} \cup SUB(x)$;
9           if $y_i \neq \$$ then
10          while $r \neq \lambda$ do
11              guess $\alpha_j \in V$;
12              if $x_j \backslash y_i \downarrow$ then $r$ := $x_j \backslash y_i$; $A(\tau,\alpha_i) \leftarrow \alpha_j$
13                                else reject fi;
14          od; fi;
15          if $A(\tau,\alpha_i) \downarrow$ then $i$ := $i+1$ else reject fi;
16      od;
17      $\bar{x}$ := $<y_1,\ldots,y_k>$;
18      CONTROL $\leftarrow \tau$;
19  od;
20  if CONTROL $\downarrow$ then accept else reject fi
```

Figure 3.1.

In order to decide whether $\bar{x} \overset{\tau}{\vdash} <y_1,\ldots,y_k>$ the algorithm guesses a substitution τ and substrings y_i $(1 \leq i \leq k)$ of x. As pointed out, storing these substrings only requires $\mathcal{O}(\log n)$ space. Similarly, we can store the variable r and execute the test $x_j \backslash y_i \downarrow$ (line 12) within space $\mathcal{O}(\log n)$. Note that in lines 10-14 we determine whether $y_i = h_{\bar{x}}(w)$ for some w over V (cf. Definition 3.1). The next step consists of deciding whether w is a word in the language $\tau(\alpha_i)$ (cf. lines 12 and 15). This can be resolved using $\mathcal{O}(S(n))$ space, since $\tau(\alpha_i) \in K$ and by assumption K is included in 1-NSPACE(S(n)).

From the assumption that $S(n) \geq \log n$, it follows that the total amount of space required for the execution of the algorithm is $\mathcal{O}(S(n))$.

Thus $L(G) \in \mathrm{NSPACE}(S(n))$ and hence $\eta(\Gamma,K) \subseteq \mathrm{NSPACE}(S(n))$.
(2) If we omit line 18 in the algorithm of Figure 3.1 and if we replace line 20 by

```
20 accept
```

then we obtain in an analogous way the corresponding result for the uncontrolled case. □

COROLLARY 3.4. *Let* $S(n) \geq n$, *and let* Γ *and* K *be families satisfying the assumptions of Lemma* 2.3.
(1) *If both* Γ *and* K *are included in* $\mathrm{NSPACE}(S(n))$, *then* $\eta(\Gamma,K) \subseteq \mathrm{NSPACE}(S(n))$.
(2) *If* $K \subseteq \mathrm{NSPACE}(S(n))$, *then* $\eta(K) \subseteq \mathrm{NSPACE}(S(n))$.

Proof. Both statements easily follow from Theorem 3.3 and the fact that 1-NSPACE(S(n)) equals NSPACE(S(n)) for $S(n) \geq n$. □

To obtain the lower bound of log n in Theorem 3.3 we use in an essential way configurations instead of complete sentential forms, since storing complete sentential form requires $\mathcal{O}(n)$ space. In order to obtain a result similar to Corollary 3.4 for iterated nondeterministic substitution we have to use complete sentential forms, since due to the fact that the substitutions are nondeterministic we cannot rely on configurations. Establishing the following result (which was also mentioned in (Van Leeuwen, 1976) in an implicit way) is straightforward and the proof is therefore left to the reader.

THEOREM 3.5. *Let* $S(n) \geq n$, *and let* Γ *and* K *be families satisfying the conditions of Lemma* 2.3.
(1) *If both* Γ *and* K *are included in* $\mathrm{NSPACE}(S(n))$, *then* $H(\Gamma,K) \subseteq \mathrm{NSPACE}(S(n))$.
(2) $K \subseteq \mathrm{NSPACE}(S(n))$ *implies* $H(K) \subseteq \mathrm{NSPACE}(S(n))$. □

Van Leeuwen (1976) established an implication similar to Theorem 3.5(2) for DSPACE(S(n)) provided that $S(n) \geq n \log n$. For the sake of completeness we recall this result (Theorem 3.6(2)) together with the obvious controlled variant (Theorem 3.6(1)).

THEOREM 3.6. *Let* $S(n) \geq n \log n$, *and let* Γ *and* K *be families satisfying*

the assumptions of Lemma 2.3.
(1) *If both* Γ *and* K *are included in* DSPACE(S(n)), *then* $H(\Gamma,K) \subseteq$ DSPACE(S(n)).
(2) *If* K $\subseteq$ DSPACE(S(n)), *then* H(K) $\subseteq$ DSPACE(S(n)). □

Replacing the complete sentential forms in Van Leeuwen's proof of Theorem 3.6(2) by configurations yields the following result with respect to iterated deterministic substitution.

THEOREM 3.7. *Let* $S(n) \geq n$, *and let* Γ *and* K *be families satisfying the conditions of Lemma* 2.3.
(1) *If both* Γ *and* K *are included in* DSPACE(S(n)), *then* $\eta(\Gamma,K) \subseteq$ DSPACE(S(n)).
(2) K $\subseteq$ DSPACE(S(n)) *implies* $\eta(K) \subseteq$ DSPACE(S(n)).

Proof (sketch). A modification of Van Leeuwen's algorithm (cf. Van Leeuwen, 1976 Theorem 5.2 for details) based on configurations instead of sentential forms is straightforward and it is therefore omitted. We only analyse the effect on the space complexity due to this modification.

Enumerating sequences of substitutions still requires $\mathcal{O}(n)$ space. Storing an activation-record for each recursive call of DERIV now requires $\mathcal{O}(\log n)$ instead of $\mathcal{O}(n)$ space. Since the recursion depth remains $\mathcal{O}(\log n)$, executing the recursive procedure DERIV requires at most $\mathcal{O}((\log n)^2) + \mathcal{O}(S(n))$. For a possible test for membership in the control language we need $\mathcal{O}(n)$ space by Lemma 2.3(1).

Thus the total amount of space required is $\mathcal{O}(n) + \mathcal{O}((\log n)^2) + \mathcal{O}(S(n))[+\mathcal{O}(S(n))$ in the controlled case] which is $\mathcal{O}(S(n))$. □

In Section 4 we will see that an improvement of the lower bounds in 3.4(2)-3.7(2) to $S(n) \geq \log n$ is as hard as the $\mathcal{P} = \mathcal{NP}$ question.

4. APPLICATION

In this section we discuss some applications of the results that we proved in the previous section. We first consider closure properties of the families NSPACE(S(n)) and DSPACE(S(n)).

THEOREM 4.1. *Let* $S(n) \geq n$ *and* $S(2n) \leq c.S(n)$ *for some constant* c. *Then* NSPACE(S(n)) *and* DSPACE(S(n)) *are AFL's closed under intersection and*

and iterated λ-free deterministic substitution. Moreover, NSPACE(S(n)) *is closed under iterated λ-free nondeterministic substitution; this also holds for* DSPACE(S(n)) *provided that* $S(n) \geq n \log n$.

Proof. From 3.4(2), 3.5(2), 3.6(2) (i.e. Theorem 5.2 of Van Leeuwen, 1976) and 3.7(2) with K = NSPACE(S(n)) or K = DSPACE(S(n)), it follows that $\eta(K) \subseteq K$ and $H(K) \subseteq K$. So under the appropriate assumptions on S(n), NSPACE(S(n)) and DSPACE(S(n)) are closed under iterated λ-free nondeterministic and deterministic substitution. Closure under the latter operation implies closure under union, concatenation, Kleene + and λ-free homomorphism (Asveld and Engelfriet, 1977). Finally, closure inder intersection and the two remaining AFL-properties (viz. inverse homomorphism and intersection with regular languages) easily follows from standard constructions in automata theory which are left to the reader. □

Comparing Theorem 4.1 with the main results in (Book et al. 1970) shows that our conditions on S(n) to obtain AFL's from space-bounded complexity classes are much weaker than those in (Book et al. 1970). However in order to achieve closure under "S(n)-bounded erasing homomorphism" additional assumptions on S(n), as required in (Book et al. 1970), seems to be inevitable.

Taking S(n) in Theorem 4.1 equal to specific concrete space bounds yields the closure under η and H of some well-known language families.

COROLLARY 4.2. *The following language families are AFL's closed under intersection and under iterated λ-free deterministic substitution:*

(1) *PSPACE*

(2) *the family of nondeterministic context-sensitive languages*

(3) *the family of deterministic context-sensitive languages*

(4) *the family of two-way nondeterministic nonerasing stack automaton languages*

(5) *the family of two-way deterministic nonerasing stack automaton languages.*

Moreover the families under (1), (2), (4) *and* (5) *are also closed under iterated λ-free nondeterministic substitution.*

Proof. These properties directly follow from Theorem 4.1 and the equality of the families under (2), (3), (4) and (5) with NSPACE(n), DSPACE(n), NSPACE(n^2) and DSPACE(n log n) respectively; cf. e.g. (Hopcroft and Ullman, 1969). □

Although both DSPACE(n) and NSPACE(n) are closed under η, the problem whether DSPACE(n) is closed under H seems to be related to the LBA-problem; cf. the discussion in (Van Leeuwen, 1976). If DSPACE(n) is not closed under H, then obviously DLBA $\subsetneq$ NLBA (i.e. DSPACE(n) $\subsetneq$ NSPACE(n)). On the other hand if there is a proof showing that DSPACE(n) is indeed closed under H, then that proof could probably be modified (along the lines implicitly suggested in Wood, 1976) in order to show DLBA = NLBA.

From Theorem 3.5 and a straightforward induction it also follows that by iterating control on ETOL or EDTOL systems (in the sense of Asveld and Van Leeuwen, 1975; Engelfriet, 1978) we do not leave the family of context-sensitive languages; cf. Asveld and Van Leeuwen, 1975; Engelfriet, 1978.

Next we show that a generalization of Theorem 4.1 and Corollary 4.2 to closure under controlled iterated λ-free (non)deterministic substitution is impossible.

A family K is closed under *removal of right endmarker* if for each language $L \subseteq \Sigma^*$ and each symbol $\$ \notin \Sigma$, L\$ in K implies L in K.

Let RE be the family of recursively enumerable languages.

PROPOSITION 4.3. *Let* K *be a family closed under removal of right endmarker. If* DSPACE(log n) $\subseteq$ K $\subsetneq$ RE, *then* K *is not closed under controlled iterated* λ*-free (non)deterministic substitution. In particular this applies to each complexity class which includes* DSPACE(log n).

Proof. Let Hôm(K) be defined by Hôm(K) = {h(L) | L $\in$ K; h is a (possibly erasing) homomorphism}. Then Hôm(DSPACE(log n)) $\subseteq$ Hôm(K) $\subseteq$ Hôm(RE) = RE.

Since the Dyck languages are in DSPACE(log n) (Ritchie and Springsteel, 1972) and Hôm(DSPACE(log n)) is a full AFL (Ginsburg et al. 1969b) the Chomsky-Schützenberger Theorem implies that all context-free languages are in Hôm(DSPACE(log n)). From this fact, the closure of Hôm(DSPACE(log n)) under intersection (Ginsburg et al. 1969b) and e.g. Theorem 1 in (Baker and Book, 1974) it follows that Hôm(DSPACE(log n)) = RE, and hence Hôm(K) = RE.

Suppose K is closed under controlled iterated λ-free (non)deterministic substitution. We will show that $\hat{\text{Hom}}(K) = RE$ implies that $RE \subseteq K$. Since this contradicts $K \subsetneq RE$, the result follows.

The argument is a slight modification of the proof of Theorem 2.2 in (Asveld, 1977).

Let $L \subseteq \Sigma^*$ be in RE. Then for $\$ \notin \Sigma$ the language $L\$$ is also in RE. So there exist a language $M \subseteq U^*$ in K and a (possibly erasing) homomorphism $h: U^* \to (\Sigma \cup \{\$\})^*$ such that $h(M) = L\$$ and $\$ \notin U$. For each symbol τ in U we define a λ-free K-substitution τ over $\Sigma \cup \{\$\}$ by

$$\tau(\$) = \{h(\tau)\$\}$$

$$\tau(\alpha) = \{\alpha\} \qquad \text{for each } \alpha \in \Sigma.$$

It is easy to see that $M(\$) = L\$$. But if this language is in K, then L is in K too, i.e. $RE \subseteq K$. $\square$

From Proposition 4.3 it follows that the families mentioned in Corollary 4.2 as well as DSPACE(log n), NSPACE(log n), P and NP are not closed under controlled iterated λ-free (non)deterministic substitution. (There exist however language families K — even included in the family of context-sensitive languages — that satisfy $H(K,K) \subseteq K$ or $\eta(K,K) \subseteq K$; cf. Asveld and Van Leeuwen, 1975; Engelfriet, 1978). We conclude this section by considering the properties of the latter four families with respect to uncontrolled iterated λ-free (non)deterministic substitution (cf. Van Leeuwen, 1975; Book, 1972; Greibach, 1977; King and Wrathall, 1978).

<u>THEOREM 4.4.</u>

(1) *NP is an AFL closed under intersection and iterated λ-free (non)-deterministic substitution.*

(2) *Let* C *be either* DSPACE(log n), NSPACE(log n) *or* P. *Then the following propositions are equivalent.*

(i) $C = P = NP$

(ii) C *is closed under iterated λ-free nondeterministic substitution*

(iii) C *is closed under iterated λ-free deterministic substitution*

(iv) C *is closed under λ-free homomorphism.*

Proof. (1) Van Leeuwen (1975) proved that $H(\mathcal{NP}) = \mathcal{NP}$, and his argument can also be used in order to obtain $\eta(\mathcal{NP}) = \mathcal{NP}$.
(2) From (1) it follows that (i) implies both (ii) and (iii). Since each family including all singleton languages and closed under isomorphism ("renaming of symbols") and iterated λ-free (non)deterministic substitution is closed under λ-free homomorphism, (ii) as well as (iii) implies (iv).

The fact that (i) follows from (iv) has been established for $C = \mathcal{P}$ in (Book, 1972) and in (King and Wrathall, 1978) for the other two cases. □

Theorem 4.4(2) implies that an improvement of the lower bound in Corollary 3.4(2) and Theorems 3.5(2), 3.6(2), 3.7(2) and 4.1 from $S(n) \geq n$ to $S(n) \geq \log n$ is as hard as the $\mathcal{P} = \mathcal{NP}$ problem.

Acknowledgement

I am grateful to Jan van Leeuwen and Paul Vitányi for some fruitful discussions and for their comment on an earlier version of this paper.

References

ASVELD, P.R.J. (1977), Controlled iteration grammars and full hyper-AFL's, *Inform. Contr.* 34, 248-269.

ASVELD, P.R.J. (1978), Iterated Context-Independent Rewriting —— An Algebraic Approach to Families of Languages (Doctoral dissertation), Twente University of Technology, Enschede, The Netherlands.

ASVELD, P.R.J. and ENGELFRIET, J. (1977), Iterated deterministic substitution, *Acta Informatica* 8, 285-302.

ASVELD, P.R.J. and ENGELFRIET, J. (1979), Extended linear macro grammars, iteration grammars, and register programs, *Acta Informatica* 11, 259-285.

ASVELD, P.R.J. and VAN LEEUWEN, J. (1975), Infinite chains of hyper-AFL's, TW-memorandum No. 99, Twente University of Technology, Enschede, The Netherlands.

BAKER, B.S. and BOOK, R.V. (1974), Reversal-bounded multipushdown machines, *J. Comp. System Sci.* 8, 315-332.

BOOK, R.V. (1972), On languages accepted in polynomial time, *SIAM J. Computing* 1, 281-287.

BOOK, R.V., GREIBACH, S.A. and WEGBREIT, B. (1970), Time- and tape-bounded Turing acceptors and AFL's, *J. Comp. System Sci.* 4, 606-621.

DOWNEY, P.J. (1974), Developmental systems and recursion schemes, *in* Proc. Conf. on Biologically Motivated Automata Theory (IEEE), McLean Va. pp. 54-58.

ENGELFRIET, J. (1978), Three hierarchies of transducers, TW-memorandum No. 217, Twente University of Technology, Enschede, The Netherlands.

ERNI, W.J. (1977), Auxiliary pushdown acceptors and regulated rewriting systems, TR-59 Institut für angewandte Informatik und formale Beschreibungsverfahren, Universität Karlsruhe, FGR.

GINSBURG, S. (1975), "Algebraic and Automata-Theoretic Properties of Formal Languages", North-Holland, Amsterdam.

GINSBURG, S., GREIBACH, S.A. and HOPCROFT, J.E. (1969a), "Studies in Abstract Families of Languages", *Mem. Amer. Math. Soc.* 87.

GINSBURG, S., GREIBACH, S.A. and HOPCROFT, J.E. (1969b), Pre-AFL, *in* Ginsburg et al. (1969a) pp.41-51.

GREIBACH, S.A. (1977), A note on NSPACE($\log_2$ n) and substitution, *Rev. Française Automat. Informat. Rech. Operat. IT* 11, 127-132.

HARJU, T. (1977), A polynomial recognition algorithm for the EDTOL languages, *Elektron. Informationsverarbeit. Kybernetik* 13, 169-177.

HERMAN, G.T. and ROZENBERG, G. (1975), "Developmental Systems and Languages", North-Holland, Amsterdam.

HOPCROFT, J.E. and ULLMAN, J.D. (1969), "Formal Languages and Their Relation to Automata", Addison-Wesley, Reading, Mass.

JONES, N.D. and SKYUM, S. (1977), Recognition of deterministic ETOL languages in logarithmic space, *Inform. Contr.* 35, 177-181.

KING, K.N. and WRATHALL, C. (1978), Stack languages and log n space, *J. Comp. System Sci.* 17, 281-299.

RITCHIE, R.W. and SPRINGSTEEL, F.N. (1972), Language recognition by marking automata, *Inform. Contr.* 20, 313-330.

ROUNDS, W.C. (1973), Complexity of recognition in intermediate-level languages, *in* Proc. 14th Ann. IEEE Symp. Switching and Automata Theory, pp. 145-158.

ROZENBERG, G. (1973), Extensions of tabled OL-systems and languages, *Internat. J. Comp. Inform. Sci.* 2, 311-336.

SALOMAA, A. (1973), Macros, iterated substitution and Lindenmayer AFL's, DAIMI PB-18, University of Aarhus, Aarhus, Denmark. An abstract appeared *in* "L Systems" (G. Rozenberg and A. Salomaa, Eds.), pp.250-253, Lecture Notes in Computer Science 15, Springer-Verlag, Berlin-Heidelberg-New York.

SUDBOROUGH, I.H. (1977), The time and tape complexity of developmental languages, *in* "Automata, Languages and Programming, 4th Colloquium" (A. Salomaa, and M. Steinby, Eds.), Lecture Notes in Computer Science 52, Springer Verlag,Berlin-Heidelberg-New York, pp.509-523.

VAN LEEUWEN, J. (1973), F-iteration languages, Memorandum, University of California, Berkeley, Ca.

VAN LEEUWEN, J. (1975), Extremal properties of non-deterministic time-complexity classes, *in* "International Computing Symposium 1975" (E. Gelenbe and D. Potier, Eds.) pp.61-64, North-Holland, Amsterdam.

VAN LEEUWEN, J. (1976), A study of complexity in hyper-algebraic families, *in* "Automata, Languages, Development" (G. Rozenberg and A. Lindenmayer, Eds.) pp.323-333, North-Holland, Amsterdam.

WOOD, D. (1976), Iterated a-NGSM maps and Γ systems, *Inform. Contr.* 32, 1-26.