

Iterated Deterministic Substitution

Peter R.J. Asveld* and Joost Engelfriet

Department of Applied Mathematics, Twente University of Technology,
P.O. Box 217, Enschede, The Netherlands

Summary. Deterministic substitution of languages means substituting the same word (from a given language) for all occurrences of a symbol. For an arbitrary family K of languages the notion of deterministic K -iteration grammar is introduced, which is essentially the iteration of a finite number of deterministic substitutions of languages from K . The families of languages generated by these grammars are investigated.

Introduction

The operation of substituting words for symbols in a word is one of the basic notions in string processing and mathematical logic. In formal language theory it has been generalized successfully to the operation of substituting languages for symbols in a word (or a language). Suppose that the language L is substituted for the symbol σ in a word w . In the usual language-theoretic sense this means that a word from L is substituted for each occurrence of σ in w , such that different occurrences may be replaced by different words of L . In this paper we study an alternative definition of substitution, such that one word from L is chosen and substituted for all occurrences of σ in w . To distinguish this notion from the usual one we will call it “deterministic substitution” (or, shortly, d-substitution). We will investigate in particular the operation of iterated deterministic substitution. Our interest in (iterated) deterministic substitution came from two related areas of formal language theory: (i) macro grammars [11] and (ii) parallel rewriting systems or L-systems [13]. In [9] it was shown that the IO (inside-out) macro languages can be characterized as the solutions of recursive systems of equations with deterministic substitution (there called IO-substitution) as basic operation, and similarly for the OI (outside-in) macro languages and ordinary substitution (there called OI-substitution). Thus, according to the fixed point theorem, an IO language can be

* The work of the first author has been supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.)

obtained by iterating the operation of deterministic substitution an arbitrary number of times (and similarly for OI). In [4] a relationship was established between macro languages and L-systems, in particular ETOL systems and EDTOL systems. An ETOL system is basically the iteration of a finite number of finite substitutions, whereas an EDTOL system (or deterministic ETOL system) is the iteration of a finite number of homomorphisms. In order to study arbitrary iterated substitution Van Leeuwen and Salomaa introduced (in [23, 19]), as a generalization of ETOL system, the notion of a K -iteration grammar, which is the iteration of a finite number of substitutions (of languages from a given family K). We now note that it is rather easy to see that the iteration of a finite number of finite d-substitutions can be realized by an EDTOL system. Thus it seems to be natural to introduce the notion of a deterministic K -iteration grammar (being the iteration of a finite number of d-substitutions of languages from K) as the deterministic counterpart of the notion of K -iteration grammar. Several authors [23, 19, 2, 1, 18] have studied the properties of K -iteration languages, in particular closure properties. In order to investigate iterated deterministic substitution, we study in this paper deterministic K -iteration grammars and the languages they generate. It will turn out that results and proofs in the deterministic case are very similar to those in the usual (nondeterministic) case (cf. [1]), but they differ in some essential ways. We note that L-systems with deterministic substitutions have been studied earlier in [3, 15, 24].

This paper is divided into 5 sections. In the first section we give some preliminary definitions, in particular that of deterministic substitution. In Section 2 we introduce the basic notion of a deterministic K -iteration grammar (where K is a family of languages). The family of languages generated by these grammars is denoted by $\eta(K)$, and we say that K is closed under iterated deterministic substitution if $\eta(K) \subseteq K$. It is shown that $\eta(\text{FIN}) = \text{EDTOL}$. Moreover we define the notion of a controlled deterministic K -iteration grammar, in which the application of the d-substitutions is prescribed by a given control language. Apart from some interest on its own, control is used to facilitate proofs for the uncontrolled case. This is made possible by the fact that the use of regular control does not change the generating power of deterministic K -iteration grammars. For a given family Γ of control languages and a given K , the family of languages generated by deterministic K -iteration grammars with a control language from Γ is denoted by $\eta(\Gamma, K)$.

Section 3 contains a proof of the fact that, under some weak restrictions on K (and the control), each (controlled) deterministic K -iteration grammar is equivalent to one with λ -free d-substitutions, i.e. one in which no symbol ever generates the empty word. Such grammars are called propagating in L-system theory.

In Section 4 we study closure properties of $\eta(K)$ and $\eta(\Gamma, K)$. It is shown that, under certain restrictions on Γ and K , $\eta(\Gamma, K)$ and $\eta(K)$ are closed under iterated d-substitution and under all AFL operations except inverse homomorphism. This holds in particular for $\eta(\text{FIN}) = \text{EDTOL}$. At the end of the section we consider an inclusion diagram of several families $\eta(K)$ together with their nondeterministic counterparts, and in particular IO and OI macro languages.

In the last section we investigate the particular case of iterated d-substitution of context-free languages, i.e. the family $\eta(\text{CF})$. It is shown that languages with a certain structural property cannot be in $\eta(\text{CF}) - \eta(\text{FIN})$. This result is used to prove

that $\eta(\text{CF})$ is properly included in IO, and that $\eta(\text{CF})$ is not closed under deterministic sequential machine mappings (whereas EDTOL is). The latter result implies that closure under deterministic sequential machine mappings is independent from closure under homomorphism, intersection with a regular language, iterated d-substitution and the regular operations (union, concatenation and star).

1. Preliminaries

We assume the reader to be familiar with the basic terminology and results of formal language theory (cf. [14, 20]), the theory of parallel rewriting (in particular L-systems, cf. [13]) and the theory of Abstract Families of Languages (cf. [12]). The aim of this section is to recall some of this terminology and to introduce a few additional and more specific concepts together with their elementary properties.

Let K be a set of sets. The union of K is denoted by $\bigcup K$, and $\bigcup K = \{x | x \in X \text{ for some } X \text{ in } K\}$. In particular $\bigcup \emptyset = \emptyset$. The empty word is denoted by λ . A language is λ -free if it does not contain λ . A mapping f such that $f(x)$ is a language for every argument x , is λ -free if $f(x)$ is λ -free for all x .

Let V_∞ be a fixed denumerable infinite set of symbols. A family of languages is any set of languages over finite subsets of V_∞ . The family ONE is defined to be the family of all singleton languages, i.e. $\text{ONE} = \{\{\omega\} | \omega \in V_\infty^*\}$. The family SYMBOL is the subfamily of ONE defined by $\text{SYMBOL} = \{\{\sigma\} | \sigma \in V_\infty\}$. The families of finite, regular, context-free and recursively enumerable languages are denoted by FIN, REG, CF and RE respectively. For the definition of the families ETOL and EDTOL, see [13] or Examples 2.3.

A substitution τ over an alphabet V is (as usual) a mapping from V into the set of languages over V , extended to words over V by $\tau(\lambda) = \{\lambda\}$ and $\tau(\sigma_1 \dots \sigma_n) = \tau(\sigma_1) \dots \tau(\sigma_n)$ (in other words $\tau(\sigma_1 \dots \sigma_n) = \{\omega_1 \dots \omega_n | \omega_i \in \tau(\sigma_i)\}$), and extended to languages L over V by $\tau(L) = \bigcup \{\tau(\omega) | \omega \in L\}$. If K is a family of languages and, for each $\sigma \in V$, $\tau(\sigma) \in K$, then τ is called a K -substitution. For instance, a ONE-substitution is a homomorphism, a FIN-substitution is a finite substitution and a REG-substitution is a regular substitution. In this paper substitution will also be called “nondeterministic substitution” in order to distinguish it from its deterministic counterpart which will now be defined.

1.1. Definition. Let K be a family of languages and V an alphabet. A *deterministic K-substitution* (abbreviated by dK-substitution, or, whenever K is understood, simply by d-substitution) over V is a mapping τ from V into K such that $\tau(\sigma) \subseteq V^*$ for all $\sigma \in V$. The mapping τ is extended to languages over V as follows. For singletons $\{\omega\}$ with $\omega \in V^*$, $\tau(\{\omega\}) = \{h(\omega) | h \text{ is a homomorphism such that } h(\sigma) \in \tau(\sigma) \text{ for each } \sigma \in V\}$. For arbitrary languages over V , $\tau(L) = \bigcup \{\tau(\{\omega\}) | \omega \in L\}$. \square

Observe that, by this definition, if $\tau(\sigma) = \emptyset$ for at least one $\sigma \in V$, then $\tau(L) = \emptyset$ for all L (in the definition of $\tau(\{\omega\})$ the existence of an element in $\tau(\sigma)$ is required for each $\sigma \in V$). Otherwise we have $\tau(\{\lambda\}) = \{\lambda\}$ and $\tau(\{\sigma_1 \dots \sigma_n\}) = \{\omega_1 \dots \omega_n | \omega_i \in \tau(\sigma_i)\}$, and if $\sigma_k = \sigma_m$ then $\omega_k = \omega_m$.

Note that, in general, a dK-substitution is not a special case of a K -substitution (as its name might suggest). In fact the two notions of substitution represent two

different ways of generalizing the notion of homomorphism (which is obtained by taking K equal to ONE both in the deterministic and the nondeterministic case). As an example, let $V = \{a, b\}$ and $\tau(a) = a^*$, $\tau(b) = \{b, bb\}$. Then, with τ viewed as a dREG-substitution, $\tau(\{ababa\}) = \{a^n b a^n b a^n | n \geq 0\} \cup \{a^n b b a^n b b a^n | n \geq 0\}$; however with τ viewed as a REG-substitution, $\tau(\{ababa\}) = \{a^k x a^m y a^n | k, m, n \geq 0 \text{ and } x, y \in \{b, bb\}\}$.

We now define two particular kinds of language families.

1.2. Definition. A family K of languages is a *pseudoid* if

- (1) K contains at least one SYMBOL language,
- (2) K is closed under homomorphism, and
- (3) K is closed under intersection with regular languages. \square

Obviously each pseudoid includes the family $\text{ONE} \cup \{\emptyset\}$. Moreover, it is clear that $\text{ONE} \cup \{\emptyset\}$ is itself a pseudoid. Hence $\text{ONE} \cup \{\emptyset\}$ is the smallest pseudoid.

1.3. Definition. A family K of languages is a *Quasi Abstract Family of Languages*, abbreviated by *QAFL*, if K contains at least one SYMBOL language and K is closed under the regular operations (i.e. union, concatenation and Kleene $+$), λ -free homomorphism and intersection with regular languages. A QAFL is *full* if it is closed under arbitrary homomorphism. \square

Clearly a (full) AFL is a (full) QAFL closed under inverse homomorphism. A full QAFL is a pseudoid closed under the regular operations. Each full QAFL contains all regular languages (this would not be true if the SYMBOL condition was dropped from the definition of QAFL). Thus REG is the smallest full QAFL. Each full AFL is closed under regular substitution, however a full QAFL need not be closed under regular substitution (which would imply that it is a full AFL) or even finite substitution. In fact, EDTOL is a full QAFL which is not a full AFL since it is not closed under finite substitution (cf. [5]). It is however straightforward to show (and we shall prove it in Section 4) that EDTOL is closed under dREG-substitution. Clearly not every full QAFL is closed under dREG-substitution (REG is an obvious example). But every full QAFL is closed under dFIN-substitution: a family of languages is closed under dFIN-substitution iff it is closed under homomorphism and union (and contains \emptyset); in fact it is easy to see that a dFIN-substitution τ over V is the union of all homomorphisms h such that $h(\sigma) \in \tau(\sigma)$ for all $\sigma \in V$.

We finally define the operation of marking. A family K of languages is closed under left (right) marking, if for all $L \in K$ and all symbols σ not occurring in any word of L , the language σL ($L\sigma$ respectively) is in K . K is closed under full marking if K is closed under both left and right marking.

2. Controlled Deterministic Iteration Grammars

The notion of K -iteration grammar (where K is a family of languages) has been introduced in order to place a part of the theory of L-systems and L-languages into a more general framework [23, 19]. In fact, K -iteration grammars may be considered as ETOL-systems in which one replaces the finite substitutions (or

“tables”) by K -substitutions. From the point of view of AFL theory iteration grammars provide an attractive way of studying the operation of iterated substitution, leading to the notion of hyper-AFL.

In order to study iterated deterministic substitution we now introduce deterministic iteration grammars (i.e. iteration grammars provided with deterministic substitutions). Since, as we will show in Lemma 2.4, iteration grammars with dFIN-substitutions have the same generating power as EDTOL systems, one may view deterministic iteration grammars as the appropriate generalization of deterministic ETOL systems.

2.1. Definition. Let K be a family of languages. A *deterministic K -iteration grammar* (abbreviated by d K -iteration grammar, or, when K is understood, by d-iteration grammar) is a 5-tuple $G = (V, T, I, U, S)$ where

- V is an alphabet,
- $T \subseteq V$ (the terminal alphabet),
- I is an alphabet (the index alphabet),
- $S \in V - T$ (the initial symbol), and
- $U = \{\tau_i | i \in I\}$ is a finite set of d K -substitutions over V , indexed by I .

The *language generated by G* , denoted by $L(G)$, is defined by

$$L(G) = T^* \cap \bigcup \{ \tau_{i_n}(\dots \tau_{i_2}(\tau_{i_1}(\{S\}))) \dots | n \geq 1 \text{ and } i_j \in I \text{ for all } j, 1 \leq j \leq n \}. \quad \square$$

We shall use the following additional terminology. Let K be a family of languages. We denote by $\eta(K)$ the family of all languages generated by d K -iteration grammars; $\eta(K)$ is called the *deterministic hyper-algebraic extension of K* (abbreviated by *dhyper-algebraic extension of K*) and an element of $\eta(K)$ is said to be *dhyper-algebraic over K* (“dhyper” might be pronounced as “diaper”). For $m \geq 1$, $\eta_m(K)$ denotes the family of all languages generated by d K -iteration grammars that have at most m d-substitutions. One should note that $\eta(K)$ does not always include K ; however, if for instance $\text{SYMBOL} \subseteq K$, then obviously $K \subseteq \eta(K)$. The family K is said to be *dhyper-algebraically closed* (or informally, *closed under iterated d-substitution*) if $\eta(K) \subseteq K$.

Definition 2.1 and the terminology above will also be used for the usual (nondeterministic) case after consistently deleting “deterministic” and “d”, and replacing η by H . Thus $H(K)$, the *hyper-algebraic extension of K* , consists of all languages generated by K -iteration grammars (which have a finite set of K -substitutions).

We now give some simple examples.

2.2. Examples. (1) Let $G = (V, T, I, U, S)$ with $V = \{S, a\}$, $T = \{a\}$, I is a singleton and U consists of the d-substitution τ defined by $\tau(S) = \{a\}$ and $\tau(a) = \{\lambda, aa, aaa\}$. Then G is a dFIN-iteration grammar and $L(G) = \{a^{2^n 3^m} | n, m \geq 0\} \cup \{\lambda\}$. Hence $L(G)$ is in $\eta(\text{FIN})$ and even in $\eta_1(\text{FIN})$. Note that if τ is viewed as a nondeterministic substitution, then G is a FIN-iteration grammar with $L(G) = a^*$.

(2) Let G be the d K -iteration grammar (V, T, I, U, S) where $V = \{S, a, \#\} \cup T$, T is disjoint with $\{S, a, \#\}$, $I = \{p, q\}$, τ_p is defined by $\tau_p(S) = \{\# a \# a S, \lambda\}$ and $\tau_p(\sigma)$

$= \{\sigma\}$ for $\sigma \neq S$, and τ_q is such that $\tau_q(a) \subseteq T^*$ and $\tau_q(\sigma) = \{\sigma\}$ for $\sigma \neq a$. Then $L(G) = \{ \# w_1 \# w_1 \# w_2 \# w_2 \dots \# w_n \# w_n \mid n \geq 1 \text{ and } w_i \in \tau_q(a) \}$. \square

To see that L-systems are special cases of iteration grammars, one should observe that $H(K \cup \{\emptyset\}) = H(K)$ under very weak conditions on K (for instance when $\text{SYMBOL} \subseteq K$, see [1]), and that $\eta(K \cup \{\emptyset\}) = \eta(K)$ for all families K (d-substitutions τ such that $\tau(\sigma) = \emptyset$ for some $\sigma \in V$ are useless and can be dropped from any d-iteration grammar without changing the generated language, cf. the remark following Definition 1.1). Thus, for instance, $H(\text{FIN}) = H(\text{FIN} - \{\emptyset\})$ and $\eta(\text{ONE} \cup \{\emptyset\}) = \eta(\text{ONE})$. It is now easy to see that the following equations hold (which may also be viewed as definitions if you are unfamiliar with L-systems).

2.3. Examples. (1) $H_1(\text{ONE}) = \eta_1(\text{ONE}) = \text{EDOL}$ and $H_1(\text{FIN}) = \text{EOL}$.

(2) $H(\text{ONE}) = \eta(\text{ONE}) = \text{EDTOL}$.

(3) $H(\text{FIN}) = \text{ETOL}$. Moreover, $H(K) = \text{ETOL}$ for each family K with $\text{FIN} \subseteq K \subseteq \text{ETOL}$. In particular $H(\text{ETOL}) = \text{ETOL}$, i.e. ETOL is closed under iterated substitution [2]. It is also known that $H_m(\text{FIN}) = \text{ETOL}$ for each $m \geq 2$ [17]. \square

Furthermore, it is easy to show the following lemma, which is one of the motivations to study d-iteration grammars.

2.4. Lemma. $\eta(\text{FIN}) = \text{EDTOL}$.

Proof. By Example 2.3(2) we have to show that $\eta(\text{FIN}) = \eta(\text{ONE})$. Since obviously $\eta(\text{ONE}) \subseteq \eta(\text{FIN})$, it suffices to show that $\eta(\text{FIN}) \subseteq \eta(\text{ONE})$. To prove this, let $G = (V, T, I, U, S)$ be a dFIN-iteration grammar with $U = \{\tau_i \mid i \in I\}$. As noted before we may assume that for all τ_i and all $\sigma \in V$ $\tau_i(\sigma) \neq \emptyset$. We now construct the dONE-iteration grammar $H = (V, T, I_H, U_H, S)$, where U_H consists of all homomorphisms h over V such that, for some $\tau_i \in U$, $h(\sigma) \in \tau_i(\sigma)$ for all $\sigma \in V$. Formally we should define I_H to be

$$\{[i, \omega_1, \dots, \omega_n] \mid i \in I \text{ and } \omega_k \in \tau_i(\sigma_k) \text{ for all } k, 1 \leq k \leq n\},$$

where $V = \{\sigma_1, \dots, \sigma_n\}$, and then, for $j = [i, \omega_1, \dots, \omega_n]$ define $\tau_j \in U_H$ such that $\tau_j(\sigma_k) = \{\omega_k\}$ for all k , $1 \leq k \leq n$.

It should be obvious from the very definition of deterministic substitution (Definition 1.1) that $L(H) = L(G)$. \square

The above lemma (and its proof) illustrates a typical property of d-iteration grammars: a “finite amount” of nondeterminism that is present in a d-substitution of the grammar may be removed by replacing that d-substitution by finitely many new ones. Lemma 2.4 is for instance used implicitly in [22, 7] to obtain a “copying theorem” for ETOL and EDTOL. We note that Lemma 2.4 is not true when only one table is used: $\text{EDOL} = \eta_1(\text{ONE})$ is properly included in $\eta_1(\text{FIN})$. In fact $\eta_1(\text{FIN})$ contains all finite languages, which is not true for EDOL (cf. [13]).

In order to comply with the usual way of viewing grammars we now define the notion of derivation for d-iteration grammars. Let $G = (V, T, I, U, S)$ be a d-iteration grammar and $\phi, \psi \in V^*$. For $i \in I$, we write $\phi \xrightarrow{i} \psi$ if $\psi \in \tau_i(\{\phi\})$. For $\pi \in I^*$, we write $\phi \xrightarrow{\pi} \psi$ if $\psi \in \tau_{i_n}(\dots \tau_{i_2}(\tau_{i_1}(\{\phi\})) \dots)$, where $\pi = i_1 i_2 \dots i_n$ (in particular, if $\pi = \lambda$, then

$\phi \xRightarrow{\pi} \psi$ iff $\phi = \psi$). Finally we write $\phi \xRightarrow{*} \psi$ if $\phi \xRightarrow{\pi} \psi$ for some $\pi \in I^*$. If necessary, the symbol G is added as a subscript to \Rightarrow . As usual, a sequence ϕ_1, \dots, ϕ_n such that for all j ($1 \leq j \leq n-1$) $\phi_j \xRightarrow{i} \phi_{j+1}$ for some $i \in I$, is called a derivation from ϕ_1 to ϕ_n . Obviously $\phi \xRightarrow{*} \psi$ iff there is a derivation from ϕ to ψ . Note that, as usual, $L(G) = \{w \in T^* \mid S \xRightarrow{*} w\}$.

All these notions can be defined in exactly the same way for nondeterministic iteration grammars. For such a nondeterministic grammar one can easily prove that derivations have the following nice property (similar to context-free grammars): let $\sigma_1, \dots, \sigma_n \in V$, $\omega \in V^*$ and $\pi \in I^*$; then $\sigma_1 \sigma_2 \dots \sigma_n \xRightarrow{\pi} \omega$ iff there exist $\omega_1, \omega_2, \dots, \omega_n \in V^*$ such that $\omega = \omega_1 \omega_2 \dots \omega_n$ and $\sigma_k \xRightarrow{\pi} \omega_k$ for all k , $1 \leq k \leq n$. For d-iteration grammars a similar property can obviously be stated for one-step derivations (assuming that all $\tau_i(\sigma)$ are nonempty): let $\sigma_1, \dots, \sigma_n \in V$, $\omega \in V^*$ and $i \in I$; then $\sigma_1 \sigma_2 \dots \sigma_n \xRightarrow{i} \omega$ iff there exist $\omega_1, \dots, \omega_n \in V^*$ such that (1) $\omega = \omega_1 \dots \omega_n$, (2) $\sigma_k \xRightarrow{i} \omega_k$ for all k , $1 \leq k \leq n$, and (3) if $\sigma_k = \sigma_m$ then $\omega_k = \omega_m$ ($1 \leq k, m \leq n$). For arbitrary derivations the analogous statement is true in the only-if direction, as can be shown by an easy induction on the length of the derivation (and taking the σ 's in V^* rather than V). However, in the if-direction the statement is in general false as shown by the following simple example. Let the d-iteration grammar G have alphabet $\{a, b, c\}$ and one d-substitution τ with $\tau(a) = \tau(b) = \{c\}$ and $\tau(c) = \{a, b\}$. Then $a \Rightarrow c \Rightarrow b$ and $b \Rightarrow c \Rightarrow a$, but ab does not generate ba in two steps ($ab \Rightarrow cc \Rightarrow aa$ or $ab \Rightarrow cc \Rightarrow bb$).

As in the nondeterministic case [1], we will study derivation-controlled d-iteration grammars. Apart from being of general interest the notion of control often leads to more transparent proofs, even for the uncontrolled case. The general principle of control is simple. Instead of allowing arbitrary derivations one only allows derivations that are obtained by applying certain sequences of (d-)substitutions. These sequences are obtained from a language over the index alphabet, the so-called control language. Thus we consider d-iteration grammars with control taken from a given family of control languages.

2.5. Definition. Let Γ and K be families of languages. A *deterministic Γ -controlled K -iteration grammar* (abbreviated by d ΓK -iteration grammar) is a 6-tuple $G = (V, T, I, U, M, S)$ where (V, T, I, U, S) is a d K -iteration grammar and M is a language over I , such that $M \in \Gamma$ (the *control language* of G). The *language generated by G* , denoted by $L(G)$, is $\{w \in T^* \mid S \xRightarrow{\pi} w \text{ for some } \pi \in M\}$. \square

To exclude trivialities we shall assume in what follows that Γ contains a language containing a nonempty word.

We shall denote by $\eta(\Gamma, K)$ the family of languages generated by d ΓK -iteration grammars. For $m \geq 1$, $\eta_m(\Gamma, K)$ denotes the family of languages generated by d ΓK -iteration grammars with at most m d-substitutions.

As before, Definition 2.5 and the above notation will also be used for the nondeterministic case after deleting “deterministic” and “d”, and replacing η by H .

It should be obvious that each uncontrolled (d-)iteration grammar G may be viewed as a controlled (d-)iteration grammar with control language I^* , where I is the index alphabet of G .

2.6. Example. Consider the d K -iteration grammar G of Example 2.2(2), and let, for $i = 1, 2$, $G_i = (V, T, I, U, M_i, S)$ be the REG-controlled d K -iteration grammar with M_1

$= (pq)^*$ and $M_2 = p^*q$. Then $L(G_1) = L(G)$, but $L(G_2) = \{(\# w)^{2^n} | n \geq 1 \text{ and } w \in \tau_q(a)\}$. \square

One should observe some obvious properties of controlled (d-)iteration grammars. First, if Γ is closed under, for instance, finite substitution, then $\eta(\Gamma, K \cup \{\emptyset\}) = \eta(\Gamma, K)$: in a given control language one simply disregards all i such that $\tau_i(\sigma) = \emptyset$ for some σ by applying a finite substitution f such that $f(i) = \emptyset$ for those i . Secondly, it is clear that $H(\Gamma, \text{ONE}) = \eta(\Gamma, \text{ONE})$; these are the Γ -controlled EDTOL languages of [1]. Finally, if Γ is closed under finite substitution, then $\eta(\Gamma, \text{FIN}) = \eta(\Gamma, \text{ONE})$. In fact given a Γ -controlled dFIN-iteration grammar G one first gets rid of all “empty d-substitutions” by disregarding certain words in the control language of G , as noted above. Then one changes G into H as indicated in the proof of Lemma 2.4, and applies a finite substitution f to its control language, where, for $i \in I$, $f(i)$ is the set of all $[i, \omega_1, \dots, \omega_n]$ that are in I_H (for notation, see the proof of Lemma 2.4).

We conclude this section with a few facts which are true both for the (controlled) hyper-algebraic and dhyper-algebraic extensions. We will not give the proofs for these results because they are entirely analogous to those in the nondeterministic case, given in [1]. The first result shows that, under some simple conditions on K , regular control does not increase the generating power of (d) K -iteration grammars. This result enables us to use control in proofs concerning uncontrolled (d-)iteration grammars. In what follows, a one-to-one SYMBOL-substitution will be called an isomorphism.

2.7. Theorem. *Let K be a family of languages closed under isomorphism, such that $\text{ONE} \subseteq K$. Then $\eta(\text{REG}, K) = \eta(K)$ and $H(\text{REG}, K) = H(K)$.* \square

The next theorem is concerned with the recursively enumerable languages, obtained as (d)hyper-algebraic extension.

2.8. Theorem. *Let Γ and K be families of languages such that (1) $\{h(M) | M \in \Gamma \text{ and } h \text{ is a homomorphism}\} = \text{RE}$, and (2) $\text{ONE} \subseteq K \subseteq \text{RE}$. Then $\eta(\Gamma, K) = H(\Gamma, K) = \text{RE}$.* \square

Finally we state a result concerning the number of (d-)substitutions needed.

2.9. Theorem. *Let K be a family of languages containing at least one SYMBOL language and closed under isomorphism. Let Γ be a family of languages closed under λ -free homomorphism. Then $\eta_2(\Gamma, K) = \eta(\Gamma, K)$ and $\eta_2(K) = \eta(K)$, and $H_2(\Gamma, K) = H(\Gamma, K)$ and $H_2(K) = H(K)$.* \square

Note that in general the number of d-substitutions cannot be reduced to one; for instance, $H_1(\text{ONE}) = \eta_1(\text{ONE}) = \text{EDOL}$ is properly included in $\text{EDTOL} = \eta(\text{ONE}) = H(\text{ONE})$, cf. [16].

3. How to Obtain Propagating Grammars

In this section we will prove that (under weak conditions on Γ and K , in particular when Γ is not present) for each d ΓK -iteration grammar there exists a λ -free d ΓK -iteration grammar (i.e. one with only λ -free d-substitutions) generating the same

language except for the empty word. In terms of L-systems this means that each λ -free language in $\eta(\Gamma, K)$ can be generated by a propagating d-iteration grammar.

We note that this result is used in Section 5, but not in Section 4. For the analogous result in the nondeterministic case see [17, 23, 19, 18, 1].

We first need some additional notation and a lemma. Let ω be a word over V . Then $\text{alph}(\omega)$ denotes the set of all symbols occurring in ω . Note that $\text{alph}(\lambda) = \emptyset$. The function alph is extended to languages L by $\text{alph}(L) = \bigcup \{\text{alph}(\omega) \mid \omega \in L\}$.

3.1. Lemma. *Let Γ be closed under finite substitution and let K be closed under intersection with regular languages. Then for each $d\Gamma K$ -iteration grammar G there exists a $d\Gamma K$ -iteration grammar $H = (V, T, I, U, M, S)$ such that $L(H) = L(G)$ and: for each $\tau \in U$ and $\sigma \in V$, if ω_1 and ω_2 are in $\tau(\sigma)$, then $\text{alph}(\omega_1) = \text{alph}(\omega_2)$.*

Proof. The proof will be similar to that of Lemma 2.4. Indeed, as mentioned in the remark following that lemma, a finite amount of nondeterminism (which here is the choice of an $\omega \in \tau(\sigma)$ with a specific $\text{alph}(\omega)$) may be removed by replacing each d-substitution by finitely many new ones. Let $G = (V, T, I_0, U_0, S)$ with $U = \{\tau_i \mid i \in I_0\}$. For each $X \subseteq V$ we define the regular language $R(X) = \{\omega \in V^* \mid \text{alph}(\omega) = X\}$. Note that $R(\emptyset) = \{\lambda\}$. Now, for each language L over V , $L = \bigcup \{L \cap R(X) \mid X \subseteq V\}$. Thus each language $\tau(\sigma) \in K$ ($\tau \in U_0, \sigma \in V$) may be divided into parts of the form $\tau(\sigma) \cap R(X)$ that are again in K . The d-substitutions of H are now obtained by replacing each dK-substitution τ of G by all dK-substitutions that can be made by taking all possible combinations of the parts of the $\tau(\sigma)$, $\sigma \in V$. More precisely, we define I to be $\{[i, X_1, \dots, X_n] \mid i \in I_0 \text{ and } X_k \subseteq V \text{ for all } k, 1 \leq k \leq n\}$, where $V = \{\sigma_1, \dots, \sigma_n\}$. Then we define, for $j = [i, X_1, \dots, X_n]$, $\tau_j \in U$ such that $\tau_j(\sigma_k) = \tau_i(\sigma_k) \cap R(X_k)$. Finally we define the new control language M to be $f(M_0)$ where f is the finite substitution given by $f(i) = \{[i, X_1, \dots, X_n] \mid X_k \subseteq V, 1 \leq k \leq n\}$ for each $i \in I$. It should now be clear that $L(H) = L(G)$, and a formal proof is left to the reader. \square

We now show how to obtain λ -free grammars.

3.2. Theorem. *Let K be a pseudoid and let Γ be closed under finite substitution and right marking. Then for each $d\Gamma K$ -iteration grammar there exists a λ -free $d\Gamma K$ -iteration grammar generating the same language modulo λ .*

Proof. Let G be a $d\Gamma K$ -iteration grammar, and let $H = (V, T, I, U, M, S)$ be a $d\Gamma K$ -iteration grammar satisfying the conditions of Lemma 3.1 (and such that $\tau(\sigma)$ is nonempty for all $\tau \in U$ and $\sigma \in V$). We will construct a λ -free $d\Gamma K$ -iteration grammar $G_N = (V_N, T_N, I_N, U_N, M_N, S_N)$ such that $L(G_N) = L(H) - \{\lambda\} = L(G) - \{\lambda\}$. The basic idea of the proof is the same as in the nondeterministic case, cf. [1, 18]. In each derivation according to H we remove the “unproductive” symbols, i.e. symbols finally yielding the empty word. At each intermediate stage of the derivation we guess nondeterministically that certain symbols are unproductive, we remove all occurrences of these symbols from the intermediate word and we check during the rest of the derivation that these symbols will indeed yield λ . Thus at each intermediate stage we have a set W of symbols for which we have to check unproductiveness. This alphabet W is updated as follows. Let τ be the d-substitution used in the next step of the derivation. Then we know by Lemma 3.1 (and the determinism of the grammar) that all elements of $\text{alph}(\tau(W))$ have to be in

the set W' of unproductive symbols of the next stage (and note that if $\tau(\sigma) = \{\lambda\}$ then $\text{alph}(\tau(\sigma)) = \emptyset$ so that unproductiveness of σ is indeed checked). Thus W' will in fact be of the form $X \cup \text{alph}(\tau(W))$, where X is guessed nondeterministically. This guessing is realized by splitting the d-substitution τ into a finite number of new d-substitutions τ_X for each X . The derivation is successful whenever all symbols are terminal and W is empty. The alphabet W is remembered by attaching W to all occurrences of symbols in the intermediate word. Thus each intermediate word is of the form $[\sigma_1, W][\sigma_2, W] \dots [\sigma_n, W]$ with $\{\sigma_1, \dots, \sigma_n\} \cap W = \emptyset$.

The formal construction is as follows. Let $T_N = T$, and $V_N = T \cup \{F\} \cup \{[\sigma, W] \mid \sigma \in V \text{ and } W \subseteq V\}$, where F is a new symbol, and $S_N = [S, \emptyset]$. For each dK-substitution τ_i in U ($i \in I$) and for each proper subalphabet X of V we introduce a finite number of new dK-substitutions τ_{iX} over V_N such that for $\sigma \in T \cup \{F\}$, $\tau_{iX}(\sigma) = \{\sigma\}$, and for $\sigma \in V$ and $W \subseteq V$,

$$\tau_{iX}([\sigma, W]) = \begin{cases} h_{iXW}(\tau_i(\sigma)) \cap V_N^+ & \text{provided this set is nonempty} \\ \{F\} & \text{otherwise,} \end{cases}$$

where the homomorphism h_{iXW} is defined by

$$h_{iXW}(\alpha) = \begin{cases} \lambda & \text{if } \alpha \in X \cup \text{alph}(\tau_i(W)) \\ [\alpha, X \cup \text{alph}(\tau_i(W))] & \text{otherwise.} \end{cases}$$

Let τ_i be the dK-substitution defined by

$$\begin{aligned} \tau_i([\sigma, \emptyset]) &= \{\sigma\} & \text{for } \sigma \in T, \\ \tau_i([\sigma, W]) &= \{F\} & \text{if } \sigma \notin T \text{ or } W \neq \emptyset, \\ \tau_i(\sigma) &= \{\sigma\} & \text{for } \sigma \in T \cup \{F\}. \end{aligned}$$

Now we define $I_N = \{(i, X) \mid i \in I \text{ and } X \subsetneq V\} \cup \{t\}$. Finally let g be the finite substitution such that, for $i \in I$, $g(i) = \{(i, X) \mid X \subsetneq V\}$, and let $M_N = g(M) \cdot t$.

This completes the construction of G_N . It should be clear that G_N is a λ -free dFK-iteration grammar. It remains to show that $L(G_N) = L(H) - \{\lambda\}$. The following two statements can be proved by induction on the number of steps in the derivations involved. We use $[\sigma_1 \dots \sigma_n, W]$ as an abbreviation of $[\sigma_1, W] \dots [\sigma_n, W]$.

(1) For $\omega \in V^*$, $z \in T^+$ and $\pi \in I^*$, if $\omega \xrightarrow[\pi]{H} z$, then $[h_W(\omega), W] \xrightarrow[\pi]{G_N} z$ for some $u \in g(\pi)$, where W is the set of all symbols of ω that generate λ (in the derivation $\omega \xrightarrow[\pi]{H} z$) and h_W is the homomorphism that erases all symbols of W .

(2) For $\omega \in V^*$, $W \subseteq V$, $z \in T^+$, $\pi \in I^*$ and $u \in g(\pi)$, if $[\omega, W] \xrightarrow[\pi]{G_N} z$, then $\psi \xrightarrow[\pi]{H} z$ for each $\psi \in V^*$ such that $h_W(\psi) = \omega$, where h_W is the homomorphism erasing all symbols of W .

The detailed proofs of these statements are left to the reader. It is easy to see that they imply that $[S, \emptyset]$ generates in G_N all nonempty words generated by S in H . Hence $L(G_N) = L(H) - \{\lambda\}$. \square

It should be clear that, for the proof to be constructive, the emptiness problem for K should be decidable.

We cannot obtain the same theorem for the uncontrolled case directly from Theorem 2.7 (concerning regular control), since in the proof of Theorem 2.7

essential use is made of a d-substitution that is not λ -free. However the proof of Theorem 3.2 can easily be extended to the uncontrolled case.

3.3. Theorem. *If K is a pseudoid, then for each dK -iteration grammar there exists a λ -free one generating the same language modulo λ .*

Proof. Let G be a dK -iteration grammar with index alphabet I_0 . We may view G as a controlled d-iteration grammar with control language I_0^* . Then, according to the proofs of Lemma 3.1 and Theorem 3.2, there exists a λ -free controlled dK -iteration grammar G_N generating the same language modulo λ with control language $M_N = g(f(I_0^*)) \cdot t$. Since f and g are symbol to symbol substitutions, $M_N = I_1^* t$ with $I_1 = g(f(I_0))$. One should now observe that G_N , as obtained in the proof of Theorem 3.2, has the property that a terminal word can only be obtained by application of the terminal d-substitution τ_t and cannot be changed by application of any d-substitution of G_N . Thus it should be clear that we may change M_N into $(I_1 \cup \{t\})^*$ without affecting the generated language. Hence, if G'_N is the (λ -free) dK -iteration grammar obtained from G_N by disregarding the control language M_N , then $L(G'_N) = L(G_N) = L(G) - \{\lambda\}$. \square

Since $\text{EPDOL} \subsetneq \text{EDOL}$ [16] a similar theorem does not exist for $\eta_1(K)$, i.e. for uncontrolled d-iteration grammars with only one d-substitution.

4. Closure Properties of Deterministic Iteration Languages

Similar to the nondeterministic case (cf. [23, 19, 2, 1]) we are interested in the closure properties of $\eta(K)$ and $\eta(\Gamma, K)$. In this section we show that, under certain restrictions on Γ and K , $\eta(\Gamma, K)$ is a full QAFL closed under iterated d-substitution (Theorem 4.3). As a corollary we obtain that if K is a pseudoid, then $\eta(K)$ is a full QAFL closed under iterated d-substitution, and in fact the smallest one including K (Theorem 4.4). At the end of the section we consider some specific examples of families of languages closed under iterated d-substitution.

In the nondeterministic case, an AFL closed under iterated substitution is called a hyper-AFL. Similarly, a QAFL closed under iterated d-substitution will be called a dhyper-QAFL. It is left as an exercise to the reader to show that this is equivalent to the following definition.

4.1. Definition. A family K of languages is a *full dhyper-QAFL* if

- (1) K is a pseudoid, and
- (2) K is dhyper-algebraically closed, i.e. $\eta(K) \subseteq K$. \square

It is also left to the reader to show that every full dhyper-QAFL K is closed under dK -substitution.

We first prove that, under certain restrictions on Γ and K , $\eta(\Gamma, K)$ is a pseudoid.

4.2. Lemma. *Let K be a pseudoid and let Γ be closed under finite substitution and full marking. Then $\eta(\Gamma, K)$ is a pseudoid.*

Proof. Firstly, if K contains the SYMBOL language $\{\sigma\}$, then so does $\eta(\Gamma, K)$: construct a $d\Gamma K$ -iteration grammar with initial symbol S , any control language

containing a nonempty word and $\tau(S) = \tau(\sigma) = \{\sigma\}$ for every d-substitution τ of the grammar. Secondly we show that $\eta(\Gamma, K)$ is closed under homomorphism. Let $G = (V, T, I, U, M, S)$ be a dFK-iteration grammar and $h: T^* \rightarrow T_0^*$ a homomorphism. We may obviously assume that $V \cap T_0 = \emptyset$. Construct the dFK-iteration grammar $G_0 = (V \cup T_0, T_0, I \cup \{f\}, U_0, Mf, S)$, where U_0 consists of all $\tau \in U$, extended by $\tau(\sigma) = \{\sigma\}$ for $\sigma \in T_0$, and τ_f , defined by $\tau_f(\sigma) = \{h(\sigma)\}$ for $\sigma \in T$ and $\tau_f(\sigma) = \{\sigma\}$ otherwise. Clearly $L(G_0) = h(L(G))$.

Finally we prove that $\eta(\Gamma, K)$ is closed under intersection with regular languages. Let $G = (V, T, I, U, M, S)$ be a dFK-iteration grammar, and let R be a regular language over T accepted by the deterministic finite automaton (Q, T, δ, q_0, Q_F) , where Q is the set of states, q_0 the initial state, $Q_F \subseteq Q$ the set of final states and $\delta: Q \times T \rightarrow Q$ the transition function. To obtain a grammar H generating $L(G) \cap R$ we cannot use the usual construction with triples $[q_1, \sigma, q_2]$, where $q_1, q_2 \in Q$ and $\sigma \in V$, because in general this would change two occurrences of the same symbol into two different triples, thus destroying the synchronization of these two occurrences due to the determinism of the substitutions. Instead we shall use another well known construction with pairs $[\sigma, \phi]$, where $\sigma \in V$ and $\phi: Q \rightarrow Q$ is the state transformation corresponding to the word in T^* generated by σ . Then it suffices to replace all occurrences of σ by occurrences of $[\sigma, \phi]$ for some ϕ . This is realized by replacing every d-substitution of the original grammar by finitely many new d-substitutions, one for each possible assignment of mappings ϕ to symbols $\sigma \in V$.

The formal construction is as follows. Let Φ denote the set of all mappings from Q into Q . For each $w \in T^*$ we define δ_w (the state transformation corresponding to w) as usual, such that $\delta_{w_1 w_2} = \delta_{w_2} \circ \delta_{w_1}$ (where \circ denotes usual function composition), δ_λ is the identity and, for $\sigma \in T$, $\delta_\sigma(q) = \delta(q, \sigma)$. Then $R = \{w \in T^* \mid \delta_w(q_0) \in Q_F\}$. Now construct the dFK-iteration grammar $H = (V_1, T, I_1, U_1, M_1, S_1)$ such that $V_1 = (V \times \Phi) \cup \{S_1, F\} \cup T$, where S_1 and F are new symbols, and the new d-substitutions and control language are defined in the following way. First we define for each $\phi \in \Phi$ the language R_ϕ over $V \times \Phi$ by $R_\phi = \{[\sigma_1, \phi_1][\sigma_2, \phi_2] \dots [\sigma_n, \phi_n] \mid n \geq 1 \text{ and } \phi_n \circ \dots \circ \phi_2 \circ \phi_1 = \phi\} \cup E_\phi$, where

$$E_\phi = \begin{cases} \{\lambda\} & \text{if } \phi \text{ is the identity mapping} \\ \emptyset & \text{otherwise.} \end{cases}$$

Obviously R_ϕ is a regular language. Secondly we define for each mapping $f: V \rightarrow \Phi$ the homomorphism $f': V^* \rightarrow (V \times \Phi)^*$ by $f'(\sigma) = [\sigma, f(\sigma)]$ for each $\sigma \in V$.

Now let $I_1 = \{(i, f) \mid i \in I \text{ and } f: V \rightarrow \Phi\} \cup \{\phi \in \Phi \mid \phi(q_0) \in Q_F\} \cup \{t\}$. For each $i \in I$ and $f: V \rightarrow \Phi$ the d-substitution τ_{if} is defined such that, for $\sigma \in V$ and $\phi \in \Phi$,

$$\tau_{if}([\sigma, \phi]) = \begin{cases} \{F\} & \text{if } \tau_i(\sigma) = \{\lambda\} \text{ and } \phi \text{ is not the identity} \\ f'(\tau_i(\sigma)) \cap R_\phi & \text{otherwise,} \end{cases}$$

$$\tau_{if}(\sigma) = \{\sigma\} \quad \text{for } \sigma \in \{S_1, F\} \cup T.$$

For each ϕ such that $\phi(q_0) \in Q_F$ the d-substitution τ_ϕ is defined such that $\tau_\phi(S_1) = \{[S, \phi]\}$ and $\tau_\phi(\sigma) = \{\sigma\}$ for $\sigma \neq S_1$. The d-substitution τ_t is defined by

$$\tau_t([\sigma, \phi]) = \{\sigma\} \quad \text{if } \sigma \in T \text{ and } \phi = \delta_\sigma,$$

$$\tau_t(\sigma) = \{\sigma\} \quad \text{otherwise.}$$

Finally the control language M_1 of H is defined to be $g(sM) \cdot t$, where s is a new symbol and g is the finite substitution such that

$$\begin{aligned} g(s) &= \{\phi \in \Phi \mid \phi(q_0) \in Q_F\} \quad \text{and, for } i \in I, \\ g(i) &= \{(i, f) \mid f: V \rightarrow \Phi\}. \end{aligned}$$

This completes the construction of the $d\Gamma K$ -iteration grammar H . It is left to the reader to show that $L(H) = L(G) \cap R$. \square

We now prove the main result of this section.

4.3. Theorem. *Let K be a pseudoid and let Γ be closed under concatenation, Kleene $*$, finite substitution and full marking. Then $\eta(\Gamma, K)$ is a full dhyper-QAFL.*

Proof. In Lemma 4.2 we have shown that $\eta(\Gamma, K)$ is a pseudoid. It remains to show that $\eta(\Gamma, K)$ is dhyper-algebraically closed, i.e. $\eta(\eta(\Gamma, K)) \subseteq \eta(\Gamma, K)$. The proof is similar to that for the nondeterministic case in [1]. Let $G = (V, T, I, U, S)$ be a dK' -iteration grammar with $K' = \eta(\Gamma, K)$. Let $V = \{\sigma_1, \dots, \sigma_n\}$ and let $U = \{\tau_i \mid i \in I\}$, where each τ_i is a dK' -substitution over V (and assume that the $\tau_i(\sigma_j)$ are nonempty). Let, for each $i \in I$ and each j ($1 \leq j \leq n$), ϕ_{ij} be an isomorphism (i.e. one-to-one SYMBOL-substitution) from V into T_{ij} , where the T_{ij} are new alphabets. Let $\tau_i(\sigma_j) = \phi_{ij}^{-1}(L(G_{ij}))$, where the $G_{ij} = (V_{ij}, T_{ij}, I_{ij}, U_{ij}, M_{ij}, S_{ij})$ are $d\Gamma K$ -iteration grammars. We may obviously assume that all V_{ij} are disjoint. We now construct a $d\Gamma K$ -iteration grammar H such that $L(H) = L(G)$. Let $H = (V_0, T, I_0, U_0, M_0, S)$ where $V_0 = (\bigcup_{i,j} V_{ij}) \cup T \cup \{F\}$ (with F a new symbol), $I_0 = I \cup (\bigcup_{i,j} I_{ij}) \cup \{t\}$ (with t new), $U_0 = \{\mu_i \mid i \in I_0\}$ and U_0 and M_0 are defined as follows. For each $i \in I$, μ_i is defined such that $\mu_i(\sigma_j) = \{S_{ij}\}$ for $1 \leq j \leq n$, and $\mu_i(\sigma) = \{F\}$ for $\sigma \notin V$. For each $k \in I_{ij}$, μ_k is defined by

$$\begin{aligned} \mu_k(\sigma) &= v_k(\sigma) \quad \text{for } \sigma \in V_{ij} \text{ (where } U_{ij} = \{v_k \mid k \in I_{ij}\}), \\ \mu_k(\sigma) &= \{\sigma\} \quad \text{for } \sigma \notin V_{ij}. \end{aligned}$$

The d -substitution μ_t is defined by

$$\begin{aligned} \mu_t(\sigma) &= \{\phi_{ij}^{-1}(\sigma)\} \quad \text{if } \sigma \in T_{ij} \text{ for some } i \in I \text{ and } 1 \leq j \leq n, \\ \mu_t(\sigma) &= \{F\} \quad \text{otherwise.} \end{aligned}$$

The new control language M_0 is as follows. Let $I = \{i_1, \dots, i_p\}$. Then $M_0 = (M_1^* \dots M_p^*)^* = (M_1 \cup \dots \cup M_p)^*$, where for $1 \leq s \leq p$ $M_s = i \cdot M_{i_1} \dots M_{i_n} \cdot t$ (with $i = i_s$).

This completes the construction of H . H simulates the derivations of G as follows. Intermediate words generated by H that correspond to intermediate words in a derivation of G , are written in exactly the same way (over the alphabet V). For $i \in I$, a τ_i -step of G is simulated by H under control M_s with $i = i_s$. By a single application of μ_i all symbols σ_j of V are converted into the corresponding symbols S_{ij} of G_{ij} . Then the languages M_{ij} (for j from 1 to n) control the generation of words according to the G_{ij} , and μ_t checks that these words are terminal (and thus belong to $L(G_{ij})$) and converts them back into symbols of V . Note that different occurrences of the same S_{ij} are rewritten under the same control word from M_{ij} and thus, by the

determinism of the grammar, generate the same word (as required by the determinism of τ_i). A formal proof that $L(H) = L(G)$ is left to the reader. \square

We note that Theorem 4.3 also holds when “concatenation” is replaced by “union” (the proof is slightly more complicated).

As a direct consequence of the above theorem and Theorem 2.7 on regular control we obtain the next result covering the uncontrolled case.

4.4. Theorem. *If K is a pseudoid, then $\eta(K)$ is the smallest full dhyper-QAFL including K .*

Proof. Theorems 4.3 and 2.7 imply that $\eta(K)$ is a full dhyper-QAFL. Obviously $K \subseteq \eta(K)$. Finally, let K_0 include K and be dhyper-algebraically closed. Then $K \subseteq K_0$ implies $\eta(K) \subseteq \eta(K_0)$, and since $\eta(K_0) \subseteq K_0$, this shows that $\eta(K) \subseteq K_0$. \square

From this theorem it follows that, for an arbitrary K , the smallest full dhyper-QAFL including K is $\eta(\psi(K))$, where $\psi(K)$ is the smallest pseudoid including K (in fact, $\psi(K) = \{h(L \cap R) \mid L \in K \cup \text{SYMBOL and } R \in \text{REG}\}$).

The existence of a smallest pseudoid (viz. $\text{ONE} \cup \{\emptyset\}$) now implies the existence of a smallest full dhyper-QAFL.

4.5. Corollary. *EDTOL is the smallest full dhyper-QAFL.*

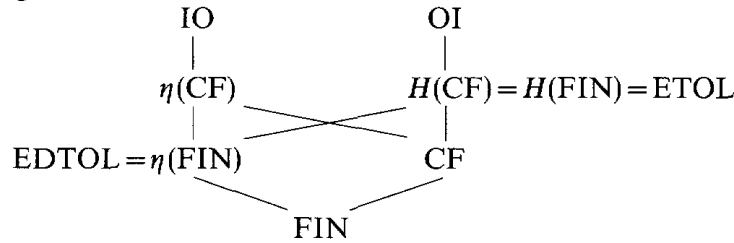
Proof. By Theorem 4.4, $\eta(\text{ONE} \cup \{\emptyset\}) = \text{EDTOL}$ is a full dhyper-QAFL. Let K be a full dhyper-QAFL. Hence K is a pseudoid and $\eta(K) \subseteq K$. Then $\text{ONE} \cup \{\emptyset\} \subseteq K$, and so $\eta(\text{ONE} \cup \{\emptyset\}) \subseteq \eta(K) \subseteq K$. \square

In the remaining part of this section we consider the relationships between a few dhyper-QAFL's and hyper-AFL's. As we have seen above, EDTOL is a full dhyper-QAFL. It is not a full hyper-AFL since it is not even closed under finite substitution. We do not know whether there is an AFL that is closed under iterated d-substitution but not under iterated substitution. An example of a hyper-AFL not closed under iterated d-substitution is ETOL. In fact ETOL is not even closed under d-substitution. Suppose to the contrary that it is. Then, for every $L \in \text{ETOL}$, $L' = \{w\#w \mid w \in L\}$ is in ETOL ($L' = \tau(\{a\#a\})$ where τ is the d-substitution with $\tau(a) = L$ and $\tau(\#) = \{\#\}$). According to [22], if L is in ETOL, then L is in EDTOL. This contradicts the existence of languages in ETOL—EDTOL.

Next we consider the macro languages of [11]. These languages are closely related to L-systems [4] and to the operations of d-substitution and substitution. Let IO and OI denote the families of inside-out and outside-in macro languages respectively. It was shown in [9] that the IO languages are the fixed point solutions of recursive systems of equations with d-substitution as basic operation, and similarly for OI and substitution respectively. Thus it should be no great surprise that IO is a full dhyper-QAFL, and that OI is a full hyper-AFL. OI being a full hyper-AFL was shown in [11, 23, 19, 4]. It is known from [11] that IO is a full QAFL. That IO is closed under iterated d-substitution can be established roughly as follows. Consider a dIO-iteration grammar $G = (V, T, I, U, S)$ with $U = \{\tau_i \mid i \in I\}$. Since IO is closed under intersection with a regular language, it suffices to show that $F(G) = \{\omega \in V^* \mid S \xRightarrow{*} \omega\}$ is in IO. Consider the recursive equation $X = \{S\} \cup (\bigcup_{i \in I} \tau_i(X))$. The fixed point solution of this equation in the powerset of T^*

is precisely $F(G)$. On the other hand, using the characterization of IO mentioned above [9], it is easy to prove that the solution of the equation is an IO language. Hence $F(G)$ is in IO. This proof is also valid for OI, and is in fact essentially the same as that for OI in [4]. We note that IO is not closed under finite substitution [9], whereas OI is not closed under d-substitution (the proof being analogous to that for ETOL, using a result of [10]).

In the next section we shall consider the full dhyper-QAFL $\eta(\text{CF})$ of languages obtained by iterated d-substitution of context-free languages. The relationships between all these families of languages can be expressed in the following inclusion diagram.



The correctness of this diagram can be shown as follows. All inclusions are well known except that between $\eta(\text{CF})$ and IO. Clearly, $\text{CF} \subseteq \text{IO}$ implies $\eta(\text{CF}) \subseteq \eta(\text{IO})$. But we just showed that IO is closed under iterated d-substitution, i.e. $\eta(\text{IO}) \subseteq \text{IO}$. Hence $\eta(\text{CF}) \subseteq \text{IO}$.

Incomparability of EDTOL and CF was proved in [6]. There is a language in $\text{ETOL} \setminus \text{IO}$ (cf. [11]). There is also a language in $\eta(\text{CF}) \setminus \text{OI}$: if $L \in \text{CF} \setminus \text{EDTOL}$, then $\{w\# w\# w \mid w \in L\}$ is obviously in $\eta(\text{CF})$, but not in OI [10]. The proper inclusion of ETOL in OI was shown in [7]. Finally, the proper inclusion of $\eta(\text{CF})$ in IO will be proved in the next section.

5. Iterated Deterministic Substitution of Context-Free Languages

In this section we consider the particular dhyper-QAFL $\eta(\text{CF})$. Using the pumping lemma for context-free languages, it will be shown (in Theorem 5.5) that languages having a certain structural property cannot be generated by dCF-iteration grammars, unless they could already be generated by a dFIN-iteration grammar (i.e. an EDTOL system). This result is similar to that in [2, 21] concerning $H_1(\text{REG})$ and $H_1(\text{FIN})$, using the pumping lemma for regular languages (cf. also the “copying theorems” of [22, 10]). It will be used for two purposes. First to show that $\eta(\text{CF})$ is properly contained in IO (Corollary 5.7). Secondly, since EDTOL is closed under deterministic gsm mappings (see [1, 8]), one might suspect that every full dhyper-QAFL is closed under deterministic gsm mappings (generalizing for instance the proof of Lemma 4.2). It turns out, however, that the operation of deterministic gsm mapping is independent of the dhyper-QAFL operations. In fact, $\eta(\text{CF})$ is not closed under deterministic sequential machine mappings (Corollary 5.6).

For later use we first state without proof two basic lemmas about derivation in a d-iteration grammar (cf. the remarks in Section 2). The first lemma shows that, as usual, a derivation can be cut into parts. The second lemma shows that derivations

can be added to a given derivation, provided they were already part of it (for instance, if $\sigma_1 \sigma_2 \xrightarrow{*} \omega_1 \omega_2$, then $\sigma_1 \sigma_2 \sigma_2 \sigma_1 \xrightarrow{*} \omega_1 \omega_2 \omega_2 \omega_1$). Let, in these lemmas, $G = (V, T, I, U, S)$ be a d-iteration grammar.

5.1. Lemma. *Let $\sigma_1, \dots, \sigma_k \in V$ ($k \geq 1$), $\omega \in V^*$ and $\pi \in I^*$. If $\sigma_1 \dots \sigma_k \xrightarrow{\pi} \omega$, then there are $\omega_1, \dots, \omega_k \in V^*$ such that $\omega = \omega_1 \dots \omega_k$, $\sigma_j \xrightarrow{\pi} \omega_j$ for $1 \leq j \leq k$, and if $\sigma_i = \sigma_j$ then $\omega_i = \omega_j$. \square*

5.2. Lemma. *Let $\sigma_1, \dots, \sigma_k \in V$, $\omega_1, \dots, \omega_k \in V^*$ ($k \geq 1$) and $\pi \in I^*$. If $\sigma_1 \dots \sigma_k \xrightarrow{\pi} \omega_1 \dots \omega_k$ and $\sigma_j \xrightarrow{\pi} \omega_j$ for $1 \leq j \leq k$, then $\sigma_{j_1} \sigma_{j_2} \dots \sigma_{j_n} \xrightarrow{\pi} \omega_{j_1} \omega_{j_2} \dots \omega_{j_n}$ for all j_s ($1 \leq s \leq n$) such that $1 \leq j_s \leq k$. \square*

We now formulate the above mentioned structural property of languages.

5.3. Definition. Let L be a language over alphabet T . L has *property P* if the following holds: For all $n \geq 1$ and all $u, v, x_0, x_1, \dots, x_{2n} \in T^*$, if $x_0 u^k x_1 v^k x_2 u^k x_3 v^k x_4 \dots x_{2n-2} u^k x_{2n-1} v^k x_{2n} \in L$ for all $k \geq 0$, then $u = v = \lambda$. \square

As an example we show that languages containing only words that consist of three “differently coloured” parts that determine each other, have property *P*.

5.4. Example. Let T_1, T_2 and T_3 be mutually disjoint alphabets, $\#$ a new symbol, and $f: T_1^* \rightarrow T_2^*$ and $g: T_1^* \rightarrow T_3^*$ one-to-one mappings. Then, for any L over T_1 , $L = \{w \# f(w) \# g(w) \mid w \in L\}$ has property *P*. (*Proof.* Suppose that $\phi_1 = x_0 u x_1 v x_2 \dots x_{2n-2} u x_{2n-1} v x_{2n}$ and $\phi_2 = x_0 u^2 x_1 v^2 x_2 \dots$ are both in L , and that $u v \neq \lambda$. Then clearly, u and v cannot contain $\#$. Thus $u v$ contains symbols of at most two of the alphabets T_1, T_2, T_3 . Hence ϕ_1 and ϕ_2 have (at least) one part the same, but also (at least) one part different. This contradicts the bijectivity of f and g .) \square

The next theorem is the main result of this section.

5.5. Theorem. *Let L be a language with property *P*. If $L \in \eta(\text{CF})$ then $L \in \text{EDTOL}$.*

Proof. Let $G = (V, T, I, U, S)$ be a dCF-iteration grammar generating L . Let $U = \{\tau_i \mid i \in I\}$. We have to show that $L(G) \in \text{EDTOL} = \eta(\text{FIN})$. We assume that G is λ -free, i.e. has only λ -free substitutions (Theorem 3.3). Consider all context-free languages $\tau_i(\sigma)$, $i \in I$ and $\sigma \in V$. Let q be an integer such that if $z \in \tau_i(\sigma)$ and $|z| \geq q$, then $z = z_1 s z_2 t z_3$ such that $st \neq \lambda$ and $z_1 s^k z_2 t^k z_3 \in \tau_i(\sigma)$ for all $k \geq 0$ ($|z|$ denotes the length of z). The existence of such a q is guaranteed by the pumping lemma for context-free languages. In what follows we will show that no words of length $\geq q$ from any $\tau_i(\sigma)$ are ever used in a derivation of a word from $L(G)$. This implies that $L(G) = L(G')$ where G' is obtained from G by changing each $\tau_i(\sigma)$ into $\{z \in \tau_i(\sigma) \mid |z| < q\}$. Hence $L(G)$ is generated by a dFIN-iteration grammar and so $L(G) \in \eta(\text{FIN}) = \text{EDTOL}$.

Suppose to the contrary that $z \in \tau_i(\sigma)$ with $|z| \geq q$ is used in a derivation $S \xrightarrow{*} w$, $w \in T^*$. Thus there are $\phi, \psi \in V^*$ and $\pi \in I^*$ such that $S \xrightarrow{*} \phi \xrightarrow{i} \psi \xrightarrow{\pi} w$, $\phi = \phi_1 \sigma \phi_2 \sigma \phi_3 \dots \phi_r \sigma \phi_{r+1}$ ($r \geq 1$; the ϕ_j do not contain σ) and $\psi = \psi_1 z \psi_2 z \psi_3 \dots \psi_r z \psi_{r+1}$, where $\phi_j \xrightarrow{\pi} \psi_j$ for all j , $1 \leq j \leq r+1$. By the choice of q we have that $z = z_1 s z_2 t z_3$ with $st \neq \lambda$ and $z_1 s^k z_2 t^k z_3 \in \tau_i(\sigma)$ for all $k \geq 0$. Thus $\psi = \psi_1 z_1 s z_2 t z_3 \psi_2 \dots \psi_r z_1 s z_2 t z_3 \psi_{r+1}$. Since $\psi \xrightarrow{\pi} w$, it follows from Lemma 5.1 that w is of the form $w = w_1 y_1 u y_2 v y_3 w_2 \dots w_r y_1 u y_2 v y_3 w_{r+1}$, where $\psi_j \xrightarrow{\pi} w_j$

($1 \leq j \leq r+1$), $z_j \xrightarrow{\pi} y_j$ ($j=1, 2, 3$), $s \xrightarrow{\pi} u$ and $t \xrightarrow{\pi} v$. We are now going to change the derivation $S \xrightarrow{*} w$ by using $z_1 s^k z_2 t^k z_3$ (for any k) rather than z in the derivation step by τ_i , as follows:

$$\begin{aligned} S &\xrightarrow{*} \phi \xrightarrow{i} \psi_1 z_1 s^k z_2 t^k z_3 \psi_2 \dots \psi_r z_1 s^k z_2 t^k z_3 \psi_{r+1} \\ &\xrightarrow{\pi} w_1 y_1 u^k y_2 v^k y_3 w_2 \dots w_r y_1 u^k y_2 v^k y_3 w_{r+1} = w(k) \end{aligned}$$

(where the derivation according to π is justified by application of Lemma 5.2: several copies of the derivations $s \xrightarrow{\pi} u$ and $t \xrightarrow{\pi} v$, which were already present, are added to the original derivation). Hence, for all $k \geq 0$, $w(k) \in L(G)$. It now follows from the form of the $w(k)$ and the fact that $L(G)$ has P , that $u = v = \lambda$. And this implies that $s = t = \lambda$ by the fact that G is λ -free. This contradicts the fact that $s \neq t$. \square

As a consequence of this theorem we can immediately show that $\eta(\text{CF})$ is not closed under “colouring”.

5.6. Corollary. $\eta(\text{CF})$ is a full dhyper-QAFL not closed under deterministic sequential machine mappings.

Proof. The proof is based on the existence of a language $L_0 \in \text{CF} - \text{EDTOL}$ (proved in [6]). Clearly $L_1 = \{w \# w \# w \mid w \in L_0\}$ is in $\eta(\text{CF})$. However, $L_2 = \{w \# \bar{w} \# \bar{w} \mid w \in L_0\}$ is not in $\eta(\text{CF})$. (By Example 5.4, L_2 has property P . Hence according to Theorem 5.5, $L_2 \in \eta(\text{CF})$ would imply that $L_2 \in \text{EDTOL}$. Since EDTOL is closed under deterministic gsm mappings, $L_2 \in \text{EDTOL}$ would imply that $L_0 \in \text{EDTOL}$.) Obviously L_2 can be obtained from L_1 by a deterministic sequential machine. \square

Finally we prove the proper inclusion of $\eta(\text{CF})$ in IO (cf. the end of Section 4).

5.7. Corollary. $\eta(\text{CF})$ is properly included in IO.

Proof. In [11] it is shown that the language $L = \{(b^k a)^m b^k \mid k \geq 1, m = 2^k - 1\}$ is in IO—EDTOL. We shall show that L has property P . This implies, by Theorem 5.5, that $L \notin \eta(\text{CF})$. To show that L has property P , assume that both $\phi_1 = x_0 u x_1 v x_2 \dots x_{2n-2} u x_{2n-1} v x_{2n}$ and $\phi_2 = x_0 u^2 x_1 v^2 x_2 \dots$ are in L , and $uv \neq \lambda$. Denote the number of symbols a in any word w by $|w|_a$. Suppose that $|uv|_a \neq 0$. Let $\phi_1 = (b^k a)^m b^k$ with $m = 2^k - 1$. Then $|\phi_2|_a = 2^k - 1 + n|uv|_a$. Clearly $1 \leq n|uv|_a \leq 2^k - 1$. Hence $2^k \leq |\phi_2|_a \leq 2^k - 1 + 2^k - 1 = 2^{k+1} - 2$. This contradicts the fact that $\phi_2 \in L$. So $|uv|_a = 0$. Consequently $|\phi_1|_a = |\phi_2|_a$, and so $\phi_1 = \phi_2$. Hence $u = v = \lambda$, contradiction. This shows that L has property P . \square

Conclusion

It has been shown that the theory of deterministic iteration languages is very similar to that of ordinary nondeterministic iteration languages. However, proofs in the deterministic case are often more complicated than in the nondeterministic case due to the fact that in a deterministic iteration grammar the generation of a word is not context-free because of the way several occurrences of the same symbol in a sentential form are tied together. To solve this difficulty the use of several d-substitutions in the grammar seems to be indispensable. It would be interesting, but probably hard, to obtain results about deterministic iteration grammars that have

only one d-substitution (see [15, 24, 3]). Another problem is the role of nesting in deterministic iteration grammars. (A grammar is nested if, for each d-substitution τ and each symbol σ , $\sigma \in \tau(\sigma)$.) It is not clear at all whether nesting makes life easier (as in the nondeterministic case) or more difficult.

References

1. Asveld, P.R.J.: Controlled iteration grammars and full hyper-AFLs. Twente University of Technology, Enschede, Netherlands, TW-memorandum 114, 1976 (to appear in: Inform. and Control)
2. Christensen, P.A.: Hyper-AFLs and ETOL systems. In: L Systems (G. Rozenberg, A. Salomaa, eds.), Lecture Notes in Computer Science, Vol. 15, pp. 254–257. Berlin-Heidelberg-New York: Springer 1974
3. Dassow, J.: Über quasideterministische Sprachen. Rostocker Mathematisches Kolloquium 1, 51–60, 1976
4. Downey, P.J.: Formal languages and recursion schemes. Harvard University, Cambridge (Mass.), TR 16-74, Ph.D. Thesis, 1974
5. Ehrenfeucht, A., Rozenberg, G.: Three useful results concerning L languages without interaction. In: L Systems (G. Rozenberg, A. Salomaa, eds.), Lecture Notes in Computer Science, Vol. 15, pp. 72–77. Berlin-Heidelberg-New York: Springer 1974
6. Ehrenfeucht, A., Rozenberg, G.: On some context-free languages that are not deterministic ETOL languages. University of Colorado, TR CU-CS-048-74, 1974
7. Ehrenfeucht, A., Rozenberg, G., Skyum, S.: A relationship between ETOL and EDTOL languages. Theor. Comput. Sci. 1, 325–330 (1976)
8. Engelfriet, J.: Top-down tree transducers with regular look-ahead. DAIMI PB-49, University of Aarhus, Denmark, 1975 (to appear in: Math. Systems Theory)
9. Engelfriet, J., Schmidt, E.M.: IO and OL. University of Aarhus, Denmark, DAIMI PB-47, 1975 (to appear in: J. Computer System Sci.)
10. Engelfriet, J., Skyum, S.: Copying theorems. Inform. Processing Letters 4, 157–161 (1976)
11. Fischer, M.J.: Grammars with macro-like productions. Harvard University, Cambridge (Mass.), Ph.D. Thesis, 1968
12. Ginsburg, S.: Algebraic and automata-theoretic properties of formal languages. Amsterdam: North-Holland 1975
13. Herman, G.T., Rozenberg, G.: Developmental systems and languages. Amsterdam: North-Holland 1975
14. Hopcroft, J.E., Ullman, J.D.: Formal languages and their relation to automata. Reading (Mass.): Addison-Wesley 1969
15. Kudlek, M.: Comparing several ways of context-independent parallel rewriting. In: GI – 4. Jahrestagung (D. Siefkes, ed.), Lecture Notes in Computer Science, Vol. 26, pp. 122–130. Berlin-Heidelberg-New York: Springer 1975
16. Nielsen, M., Rozenberg, G., Salomaa, A., Skyum, S.: Nonterminals, homomorphisms and codings in different variations of OL-systems. I. Deterministic systems. Acta informatica 4, 87–106 (1974)
17. Rozenberg, G.: Extension of tabled OL-systems and languages. Internat. J. Computer Inform. Sci. 2, 311–336 (1973)
18. Rozenberg, G., Wood, D.: A note on K-iteration grammars. Inform. Processing Letters 4, 162–164 (1976)
19. Salomaa, A.: Macros, iterated substitution and Lindenmayer-AFLs. University of Aarhus, Denmark, DAIMI PB-18, 1973 (an abstract appeared in: L Systems, Lecture Notes in Computer Science, Vol. 15, pp. 250–253. Berlin-Heidelberg-New York: Springer 1974)
20. Salomaa, A.: Formal languages. New York: Academic Press 1973
21. Salomaa, A.: Parallelism in rewriting systems. In: Automata, languages and programming, 2nd Colloquium (J. Loeckx, ed.), Lecture Notes in Computer Science, Vol. 14, pp. 523–533. Berlin-Heidelberg-New York: Springer 1974
22. Skyum, S.: Decomposition theorems for various kinds of languages parallel in nature. SIAM J. Computing 5, 284–296 (1976)
23. Van Leeuwen, J.: F-iteration languages. University of California, Berkeley, Memorandum 1973
24. Siromoney, R., Siromoney, G.: Parallel OL-Languages. Internat. J. Computer Math. 5A, 109–123 (1975)

Received August 20, 1976