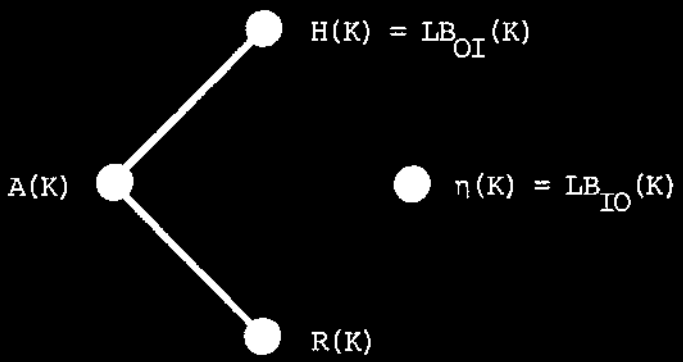
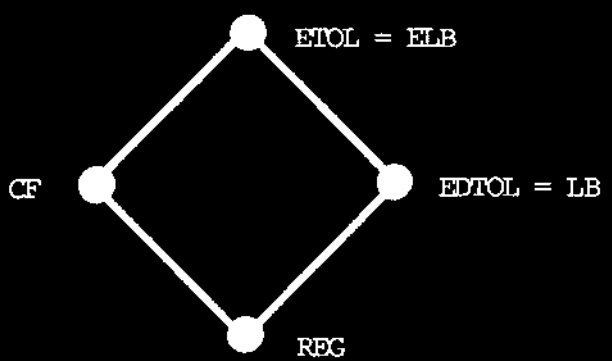
Iterated Context-Independent Rewriting



An Algebraic Approach to Families of Languages



Peter R. J. Asveld

ITERATED CONTEXT-INDEPENDENT REWRITING

AN ALGEBRAIC APPROACH TO FAMILIES OF LANGUAGES

MAY 1978

PETER R. J. ASVELD

Copyright Chapter 2 © 1977 Academic Press, New York - London

Copyright Chapter 3 © 1977 Springer-Verlag, Berlin - Heidelberg - New York

This research has been supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.)

ITERATED CONTEXT-INDEPENDENT REWRITING

AN ALGEBRAIC APPROACH TO FAMILIES OF LANGUAGES

PROEFSCHRIFT

ter verkrijging van de graad van doctor in de technische wetenschappen aan de Technische Hogeschool Twente, op gezag van de rector magnificus, prof. dr. I. W. van Spiegel, volgens besluit van het College van Dekanen in het openbaar te verdedigen op donderdag 1 juni 1978 te 16.00 uur

door

PETER ROBERT JAN ASVELD
geboren op 12 september 1947 te Enschede

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. ir. L.A.M. Verbeek

Co-referent:

Dr. J. Engelfriet

P R E F A C E

This thesis arose from my research in formal language theory during the period September 1974 to March 1978. It deals with generalized grammars (of the context-independent kind only), viz. grammars with a countable rather than a finite number of productions, satisfying the following condition: for each symbol α of the grammar the set containing all right-hand sides of productions with left-hand side equal to α, constitutes a language belonging to a given family K of languages. (So the family of languages generated by generalized grammars of a certain type obviously depends on K). In particular we study the operations on languages induced by certain types of generalized grammars and the corresponding transformations on language families.

The contents of this thesis consist of two reprinted papers and four research reports, preceded by an introduction (Chapter 1). This chapter is devoted to the basic ideas developed in this thesis and also gives a survey of the main results obtained in the subsequent chapters. It has an informal and introductory character.

The other six more formal chapters are divided into three parts.

Part I, consisting of Chapters 2, 3 and 4, deals with generalized grammars (rewriting systems). In Chapter 2 (reprinted from *Information and Control*) we investigate generalizations of (controlled) extended tabled context-independent Lindenmayer (or ETOL) systems. Chapter 3 (written with Joost Engelfriet; reprinted from *Acta Informatica*) is concerned with generalizations of (controlled) deterministic ETOL systems. In Chapter 4 (co-author: Joost Engelfriet) we investigate (controlled) extensions of linear macro grammars which enable us to characterize the language families studied in the two previous chapters and to relate them to the theory of program schemes.

Part II, containing Chapters 5 and 6, is concerned with transformations on language families, which are induced by the operations on languages

corresponding to such generalized grammars (viz. those considered in Part I and a few particularizations such as generalized context-free and regular grammars). In Chapter 5 we study the fixed points of these and related transformations and we characterize them in terms of closure properties of language families, i.e. in terms of certain types of full AFL's (AFL means Abstract Family of Languages). Chapter 6 deals with monoids generated by these transformations. The structure of these monoids provides a rather complete picture concerning (in)dependency of closure properties as studied in AFL-theory.

In Part III, i.e. Chapter 7, we use arguments from lattice theory and universal algebra in order to investigate the structure of sets of fixed points (i.e. full AFL's) belonging to the transformations defined in Part II.

These three Parts provide insight into the nature of operations on languages from an increasingly abstract point of view. It is however possible to read them (almost) independently from each other.

From the above it will be clear that readers interested in (i) grammars and rewriting systems, (ii) Abstract Families of Languages, or (iii) lattice theory and universal algebra, are referred to Part I, II and III respectively.

C O N T E N T S

A C K N O W L E D G E M E N T S

I am indebted to both my thesis advisors Joost Engelfriet and Leo Verbeek for their encouragement, help and criticism during the preparation of this thesis. Their comment considerably improved both contents and presentation of this work. Thanks are also due to the following persons who kindly provided critical remarks on preliminary versions of parts of the manuscript: Dirk Kleima (Chapter 1), Jan van Leeuwen (Chapter 2), Joel Berman and F. Loonstra (Chapter 7).

Many thanks I owe to Ria Dirks-Geerdink for the excellent way she typed this thesis.

I am grateful to Academic Press (publisher of Information and Control) and Springer-Verlag (publisher of Acta Informatica) for their permission to reprint Chapter 2 and Chapter 3 respectively.

Acknowledgements are due to the Netherlands Organization for the Advancement of Pure Research (Z.W.O.) for supporting my research during the period July 1, 1975 until December 31, 1977 (Grant No. 62-58).

Chapter 1

Iterated Context-Independent Rewriting: From Grammars to Transformations and Their Fixed Points - An Introduction

Reprinted from *TW-memorandum No. 213 (April 1978)*

The concept of rewriting or substitution, i.e. replacing (parts of) strings of symbols by other strings, is one of the central notions in mathematics, particularly in formal logic and in theoretical computer science. In certain approaches to the (constructive) foundation of mathematics, the basic principle is even that mathematics is - up to an appropriate coding - just rewriting strings of symbols (cf. e.g. [38, 40,58]) but we will by no means advocate that point of view.

Instead we restrict our attention to some rewriting mechanisms which are relevant in (theoretical) computer science, although they actually originated from other areas such as linguistics [8, 7, 3] or developmental biology [36]. In addition we set ourselves another rigorous restriction, viz. we only consider context-independent rewriting, i.e. replacing symbols in a string independently from their left or right neighbouring symbols. Although this seems to be one of the simplest ways of transforming a string of symbols into another string, a rich and still growing theory concerning this rewriting process has been developed during the last twenty years (cf.[1, 5, 18, 19, 31, 32, 41, 56]).

This thesis deals with some (rather algebraic) aspects of the theory of iterated context-independent rewriting. In particular several kinds of generalized grammars (viz. grammars with a countable rather than a finite number of rules) are investigated with emphasis on closure properties of the language families generated by these grammars.

The basic idea from which we start is the well-known observation (cf.[28,33, 42, 25, 60, 49, 50]) that the sets of productions in certain grammars, such as ETOL systems [50] and context-free grammars, may be viewed as finite substitutions. So iteratively applying productions of a grammar in order to generate a language just means iterating finite substitutions. Van Leeuwen [60, 61, 62, 63] and Salomaa [57] introduced and investigated generalized grammars, viz. grammars with a countable number of productions obtained by replacing the finite substitutions in the definition of (ordinary) grammar by K-substitutions where K is an arbitrary language family.

In the subsequent chapters we study generalizations of extended tabled context-independent Lindenmayer or ETOL systems (Chapter 2), deterministic ETOL systems (Chapter 3), linear macro grammars [17](Chapter 4), context-free and regular grammars (Chapter 5).

To illustrate the replacement of finite substitutions by K-substitutions resulting in generalized grammars, and in order to survey the main ideas developed in this thesis (particularly the results on these generalized grammars), we now consider as an example the case of context-free grammars in more detail. Context-free grammars originated from the study of natural languages [8, 7, 3] but they owe their real importance to their use in describing and analysing (parts of) the syntax of programming languages [1, 2, 5, 18, 32, 41, 56].

Remember that a context-free grammar G consists of an alphabet V, a terminal alphabet Σ ($\Sigma \subseteq V$), an initial symbol S ($S \in V-\Sigma$), and a finite set of productions (or rules) P of which each element has the form $A \to w$ with A in $V-\Sigma$ and $w \in V^*$. The set P induces a binary derivation relation $\Longrightarrow$ on V^* defined as follows: $\phi \Longrightarrow \psi$ for $\phi, \psi \in V^*$ iff $\phi = w_1Aw_2$, $\psi = w_1ww_2$ and $A \to w$ is a rule in P. As usual $\Longrightarrow^*$ denotes the reflexive and transitive closure of $\Longrightarrow$, i.e. $\phi \Longrightarrow^* \psi$ iff either $\phi = \psi$, or there exist $\phi_1, \ldots, \phi_n \in V^*$ such that $\phi_1 = \phi$,

$\phi_n = \psi$, and $\phi_i \Longrightarrow \phi_{i+1}$ for all i ($1 \leq i < n$). The context-free language L(G) generated by G is the set of all terminal words derivable from S, i.e. $L(G) = \{w \in \Sigma^* \mid S \Longrightarrow^* w\}$.

So a single step $\phi \Longrightarrow \psi$ consists of the (context-independent) replacement of one occurrence of a symbol A by the word w, whenever $A \to w$ is in P. Starting with the initial symbol S and iterating this (context-independent) rewriting process yields the set of sentential forms F(G) generated by G : $F(G) = \{w \mid S \Longrightarrow^* w\}$. The language generated by G just contains those words from F(G) which are over Σ, i.e. $L(G) = F(G) \cap \Sigma^*$.

Example 1. (context-free grammar and context-free language).
Let $G = (V,\Sigma,P,S)$ with $V = \Sigma \cup \{S,A\}$, $\Sigma = \{a,b\}$ and $P = \{S \to AA, A \to aAb, A \to \lambda\}$ where λ denotes the empty word (the string of length 0). Then $L(G) = \{a^m b^m a^n b^n \mid m,n \geq 0\}$. □

There might arise some confusion using both terms "context-free" and "context-independent". We however follow the (more or less standard) convention that "context-independent" refers to a general, vague and intuitive concept, whereas "context-free" is only used in the formal sense of context-free grammar.

We now show how the set of productions in a context-free grammar may be interpreted as a finite substitution and how it can be replaced by a K-substitution (where K refers to an arbitrary language family). First we remark that adding rules $\alpha \to \alpha$ for each (nonterminal and terminal) symbol of a context-free grammar G does not affect the language generated by G. In fact it is well-known that allowing even arbitrary productions for terminal symbols does not properly enlarge the family of context-free languages (i.e. for each context-free grammar with the additional facility that productions for terminal symbols are allowed one can easily construct an ordinary context-free grammar which still generates the same language).

Collecting for each symbol α of such a grammar G all right-hand sides of productions with left-hand side equal to α yields a finite language $\tau(\alpha)$ over the alphabet of G with the property that $\alpha \in \tau(\alpha)$. Now the generation process of the grammar G may be considered as the iterated application of the nested finite substitution τ. (The adjective "nested" refers to the fact that $\alpha \in \tau(\alpha)$ for each α of G [25]).

Before giving an example we recall the formal concept of (language-theoretic) substitution. A K-substitution τ on an alphabet V [4] is a mapping $\tau : V \to K$ extended to words over V by $\tau(\lambda) = \{\lambda\}$, $\tau(\alpha_1 \ldots \alpha_n) = \tau(\alpha_1) \ldots \tau(\alpha_n)$ for each $\alpha_1 \ldots \alpha_n \in V^+$, $\alpha_i \in V$ $(1 \leq i \leq n)$, and to languages over V by $\tau(L) = \cup \{\tau(w) \mid w \in L\}$ for each $L \subseteq V^*$. A K-substitution τ over V is a K-substitution on V such that $\tau(\alpha) \subseteq V^*$ for each α in V. A nested K-substitution on [over] V is a K-substitution on [over] V satisfying $\alpha \in \tau(\alpha)$ for each α in V. If K equals FIN, i.e. the family of finite languages, then a [nested] K-substitution [25] is called a [nested] finite substitution.

Example 2. (set of productions interpreted as nested finite substitution). Consider the context-free grammar $G = (V,\Sigma,P,S)$ of Example 1 and let τ be the nested finite substitution over V defined by $\tau(S) = \{S,AA\}$; $\tau(A) = \{A,aAb,\lambda\}$; $\tau(a) = \{a\}$; $\tau(b) = \{b\}$. Then it is easy to see that $L(G) = \cup \{\tau^n(S) \mid n \geq 0\} \cap \Sigma^*$.

Note that e.g. $\tau^2(S) = \{S,AA,\lambda,A,aAb,AaAb, aAbA,aAbaAb\}$, and in general $\tau^n(S)$ is the set of all sentential forms of G which can be obtained from S by n "nested parallel" rewriting steps (in which possibly several rules are applied simultaneously to different occurrences of symbols in a sentential form. E.g. AA derives $A,\lambda,AA,AaAb,aAbA$ and aAbaAb in one nested parallel step). □

Conversely,given a nested finite substitution τ over the alphabet V, a symbol S in V and a terminal alphabet $\Sigma \subseteq V$, we can define a context-free

grammar $G = (V,\Sigma,P,S)$ with productions $\alpha \to w$ for all $w \in \tau(\alpha)$, $\alpha \in V$. Note that G also possesses productions for terminal symbols. Due to the fact that τ is nested, the "nested parallel rewriting" of τ can easily be simulated by the usual context-free rewriting. Consequently the iterated application of τ (starting with S) produces exactly all sentential forms of G and hence $\cup \{\tau^n(S) \mid n \geq 0\} \cap \Sigma^*$ is context-free. These considerations show that context-free grammars and iterated nested finite substitutions are equivalent, and how closely grammars are related to (iterated) operations on words.

Example 3. (nested finite substitution interpreted as context-free grammar). Let $V = \Sigma \cup \{S\}$, $\Sigma = \{a,b\}$ and let the nested finite substitution τ over V be defined by $\tau(S) = \{S,b,aSa\}$, $\tau(a) = \{a\}$, and $\tau(b) = \{b,\lambda,bb\}$. Then $\cup \{\tau^n(S) \mid n \geq 0\} \cap \Sigma^* = \{a^m b^k a^m \mid m,k \geq 0\}$. An equivalent context-free grammar has for instance the productions $S \to aSa$, $S \to B$, $B \to BB$, $B \to \lambda$ and $B \to b$. □

Studying the arbitrary operation of (iterated) nested substitution (not just the finite case) now leads to the concept of generalized grammar. Replacing the nested finite substitution obtained from a context-free grammar by a nested K-substitution (for an arbitrary language family K) yields the notion of context-free K-grammar [61]. So a context-free K-grammar G is a construct (V,Σ,τ,S) where V, Σ and S are as for (ordinary) context-free grammars and τ is a nested K-substitution over V. The language generated by G is defined as $L(G) = \cup \{\tau^n(S) \mid n \geq 0\} \cap \Sigma^*$. Since each language is a countable set, a context-free K-grammar may be viewed as a context-free grammar with a countable number of productions (viz. all rules $\alpha \to w$, $w \in \tau(\alpha)$). The family of languages generated by context-free K-grammars, which obviously depends on the family K, is called the algebraic extension of K and it is denoted by A(K)[61]. Here the adjective "algebraic" refers to the context-free languages which are sometimes called algebraic languages [44].

Example 4. (context-free K-grammar).

Let CF denote the family of context-free languages. Consider the context-free CF-grammar $G = (V,\Sigma,\tau,S)$ with $V = \Sigma \cup \{S,A\}$, $\Sigma = \{a,b\}$ and τ is the nested CF-substitution defined by $\tau(S) = \{S\} \cup \{Aa^nb^n \mid n \geq 0\}$, $\tau(A) = \{A,\lambda,aAb\}$, $\tau(a) = \{a\}$ and $\tau(b) = \{b\}$. Then $L(G) = \{a^mb^ma^nb^n \mid m,n \geq 0\}$. □

Obviously, the algebraic extension of the family of finite languages equals the family of context-free languages, i.e. A(FIN) = CF. As another concrete example we note that the algebraic extension of the regular languages is also equal to the family of context-free languages i.e. A(REG) = CF where REG denotes the family of regular languages (Remember that REG is the smallest family including the finite languages, and closed under union, concatenation and Kleene *). This is actually an abstract reformulation of the well-known fact that languages generated by grammars in Extended Backus Normal Form are context-free (cf. [29]).

Above we already mentioned the close relationship between grammars and (iterated) operations. Essential in the theory of generalized grammars is the idea [60, 61, 63] that a grammar is not only a static device defining a language, but that it can also be viewed as an operation on languages. E.g. a context-free K-grammar $G = (V,\Sigma,\tau,S)$ with $V = \{\alpha_1,\ldots,\alpha_n\}$ maps the n-tuple of languages $(\tau(\alpha_1),\ldots,\tau(\alpha_n))$ into L(G) by the operation of iterated nested K-substitution. For a class of generalized grammars (of a given type) - to which corresponds some type of operations - the induced transformation on language families is called an extension. (In all cases to be considered the extension X is monotonic, i.e. $K \subseteq K'$ implies $X(K) \subseteq X(K')$ for each K and K', and indeed extensive, i.e. $K \subseteq X(K)$ for each K). Thus the class of context-free K-grammars induces the algebraic extension A(K) of K by means of $A(K) = \{L(G) \mid G \text{ is a context-free K-grammar}\} = \{\cup\{\tau^n(S) \mid n \geq 0\} \cap \Sigma^* \mid \tau \text{ is a nested K-substitution over } V;\ \Sigma \subseteq V;\ S \in V\}$.

Each extension X to be considered in the sequel has the following property: the X-extension of the family of finite languages X(FIN) equals the family generated by the class of original (ordinary) grammars on which the extension X is based (We already saw that A(FIN) = CF).

We now turn to the fixed points of such an extension of language families. Fixed points of an extension X are defined in the usual way, viz. as those elements which are invariant under X, i.e. those language families K which satisfy X(K) = K, or equivalently X(K) $\subseteq$ K.

Apart from some general mathematical interest the real importance of the fixed points belonging to an extension consists of the fact that they enable us to characterize language families which possess certain specific closure properties: they are closed under the (iterated) operation associated with the extension as discussed above. Thus, e.g. for a family K we have A(K) = K if and only if K is closed under iterated nested K-substitution (and intersection with Σ^* for each Σ). But note that fixed points have in general still other closure properties: under weak conditions on K, each family of the form A(K) shares the main closure properties with the family CF of context-free languages (because the well-known proofs for context-free grammars can be generalized in a straightforward way).

In general, under weak assumptions on K, it is often the case that for an arbitrary extension X, the family X(K) possesses the same closure properties as the family X(FIN) of languages generated by the grammars from which the extension X has been obtained (by the generalization process).

We now recall the most well-known language-theoretic closure properties, abstracted in the notion of full abstract family of languages (or full AFL). Note that in particular CF is a full AFL [20].

Let Σ_ω be an arbitrary but fixed countable set of symbols. A (language) family over Σ_ω or <u>family</u> (since Σ_ω is fixed) is any collection of languages

over finite subsets of Σ_ω. A family is nontrivial if it contains a nonempty language unequal to $\{\lambda\}$, where λ denotes the empty word. A full abstract family of languages or full AFL [20, 19, 22-26] is a nontrivial family closed under the following operations

(1) union, i.e. set-theoretical union of languages.

(2) concatenation of languages, i.e. $L_1L_2 = \{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}$.

(3) Kleene *, i.e. $L^* = \bigcup_{n\geq 0} L^n$ with $L^0 = \{\lambda\}$, $L^{i+1} = LL^i$ for $i \geq 0$.

(4) homomorphism. A homomorphism h is a ONE-substitution, where ONE is the family of all singleton languages over Σ_ω : ONE = $\{\{w\} \mid w \in \Sigma_\omega^*\}$ (Actually the notion of K-substitution has been introduced in formal language theory as a generalization of the concept of homomorphism [4]).

(5) inverse homomorphism $h^{-1}(L) = \{w \mid h(w) \in L\}$.

(6) intersection with regular languages.

Besides full AFL we need a much weaker structure called prequasoid studied in Chapter 2 as a simpler variant of the concept of quasoid introduced in [60]. It is used to formulate the "weak assumptions on K" mentioned above. A prequasoid is a nontrivial family closed under finite substitution and intersection with regular languages; a quasoid is a prequasoid containing an infinite language. Each prequasoid [quasoid] contains all finite [regular] languages and so the family FIN of finite languages [the family REG of regular languages] is the smallest prequasoid [quasoid] (Chapter 2). It is easy to show that a (pre)quasoid is a full AFL iff it is closed under REG-substitution, union, concatenation and Kleene *.

The fact that CF is a full AFL can now be generalized to the following theorem.

Theorem 1. If K is a prequasoid, then A(K) is a full AFL. □

Note that, since A(FIN) = CF and FIN is a prequasoid, Theorem 1 also covers the original case that CF is a full AFL.

An interesting situation occurs when it turns out that the family X(FIN) (of languages generated by the original grammars) itself is closed under the operations corresponding to X, i.e. XX(FIN) = X(FIN). Then we can expect that also this closure property can be generalized to X(K), i.e. XX(K) = X(K) provided as usual that K satisfies certain simple conditions.

Since CF is closed under nested iterated CF-substitution [28, 33, 42, 25] this situation actually occurs for CF and we have for the case X = A that AA(FIN) = A(FIN) = CF. This result can indeed be generalized to A(K) as follows. We provide a sketch of the proof as an example of the proof techniques involved in this field (By SYMBOL we denote the family of all singletons of length 1, i.e. SYMBOL = $\{\{\alpha\} \mid \alpha \in \Sigma_\omega\}$; an isomorphism or "renaming of symbols" is a one-to-one SYMBOL-substitution).

Theorem 2. Let K include SYMBOL and let K be closed under isomorphism and under union with SYMBOL-languages. Then A(K) is a fixed point of A, i.e. AA(K) = A(K). Moreover A(K) is the smallest fixed point of A which includes K.

Proof (sketch). Let $G = (V,\Sigma,\tau,S)$ be a context-free A(K)-grammar where τ is a nested A(K)-substitution over V. Assume that for each α in V, $G_\alpha = (V_\alpha,V,\tau_\alpha,S_\alpha)$ is a context-free K-grammar generating $\tau(\alpha)$, where τ_α is a nested K-substitution over V_α. Clearly we have to show that $L(G) \in A(K)$.

Due to the conditions on K we may assume that all alphabets $V_\alpha - V$ of nonterminals are mutually disjoint and that each G_α satisfies $\tau_\alpha(\beta) = \{\beta\}$ for each β in V.

Consider the context-free K-grammar $H = (V_0,\Sigma,\tau',S)$ where $V_0 = \cup \{V_\alpha \mid \alpha \in V\}$ and τ' is the nested K-substitution over V_0 defined by

$\tau'(\alpha) = \tau_\beta(\alpha)$ if $\alpha \in V_\beta - V$, $\beta \in V$

$\tau'(\alpha) = \{\alpha,S_\alpha\}$ if $\alpha \in V$.

It is straightforward to show that L(H) = L(G). In fact we just replaced for each α in V the countable set of rules $\{\alpha \to w \mid w \in \tau(\alpha)\}$ of G by the

rules $\alpha \to S_\alpha$ (and $\alpha \to \alpha$) and all productions of G_α. □

This theorem shows that in order to obtain the smallest family closed under iterated nested substitution, including K, one has to apply the algebraic extension only once to K (In general the smallest fixed point of A including K is $A^*(K) = \bigcup_{n\geq 0} A^n(K)$ [45]).

The fact that CF is closed under the full AFL-operations and under iterated nested CF-substitution has been abstracted in [25] to the notion of full super-AFL. A family K is a full super-AFL if K is a full AFL closed under iterated nested K-substitution. We introduce the concept of full algebraic AFL or full A-AFL to be a family which is both a prequasoid and a fixed point of the algebraic extension A. It is straightforward to show the following.

Theorem 3. A family is a full super-AFL iff it is a full A-AFL. □

Note that Theorem 3 provides a grammatical characterization of full super-AFL's because full A-AFL's are mainly defined in terms of context-free K-grammars (An automata-theoretic characterization of full super-AFL's has been given in [25] and will not be considered here).

Since each prequasoid satisfies the conditions on K in Theorem 2 we obtain the following main result as an immediate consequence of Theorems 1 and 2. We denote by $\Pi(K)$ the smallest prequasoid including K and by $\hat{\mathcal{A}}(K)$ the smallest full A-AFL (super-AFL) including K.

Theorem 4. (i) If K is a prequasoid, then A(K) is a full A-AFL, in particular it is the smallest full A-AFL including K, i.e. $\hat{\mathcal{A}}(K) = A(K)$.
(ii) $A\Pi(K)$ is a full A-AFL. In particular it is the smallest full A-AFL including K, i.e. $\hat{\mathcal{A}}(K) = A\Pi(K)$.
(iii) A(FIN) = CF is the smallest full A-AFL and is included in each full A-AFL. □

Note that Theorem 4(i) provides a grammatical characterization of the smallest full A-AFL including a prequasoid.

Theorem 4 implies in turn Theorem 1, and also Theorem 2 provided that in the latter theorem the conditions on K are replaced by the requirement that K is a prequasoid.

Finally, Theorem 4 enables us to decompose the operator $\hat{\mathcal{A}}$ into a single composition of Π and A (in that order) and therefore we refer to expressions of the form $\hat{\mathcal{A}}(K) = X\Pi(K)$ - like Theorem 4 in case X = A - as Canonical Form Theorems.

From the definition of Π and $\hat{\mathcal{A}}$ it directly follows that these transformations on language families are closure operators, i.e. both transformations are extensive, monotonic and idempotent (i.e. $\Pi\Pi(K) = \Pi(K)$ and $\hat{\mathcal{A}}\hat{\mathcal{A}}(K) = \hat{\mathcal{A}}(K)$ for each K. Consequently, $\Pi^* = \Pi$ and $\hat{\mathcal{A}}^* = \hat{\mathcal{A}}$; cf. [45]). By Theorem 2 the transformation A is also a closure operator provided we restrict the domain of A to those language families satisfying the conditions mentioned in Theorem 2. Moreover, restricted to prequasoids the operators A and $\hat{\mathcal{A}}$ coincide.

To see the relevance of a Canonical Form in obtaining e.g. the smallest full A-AFL including K, we consider the general case that we want to construct an object (viz. $\hat{\mathcal{A}}(K)$) with certain closure properties (viz. fixed point of Π and A). The general situation has been described in a very clear way in [45] from which we quote (up to a notational change in order to avoid confusion):

> ".... Suppose that we want to construct an object [viz. $\hat{\mathcal{A}}(K)$ from K] with two properties (i) it is a prequasoid, and (ii) it is a fixed point of the extension X. We find that there is a way [viz. Π] to build something with property (i), but not (ii). However, by enlarging our object [viz. from $\Pi(K)$ to $X^*\Pi(K)$], it can be made to have property (ii), though in the process, (i) is lost. But then it turns out that another enlargement

[from $X^*\Pi(K)$ to $\Pi X^*\Pi(K)$] will restore (i) while losing (ii). This would seem to be a hopeless pursuit - one that could go on forever, producing ever larger objects which have one, but not the other of our desired properties. However, in mathematics (and perhaps only in mathematics), such an infinite process may lead somewhere. By making infinitely many adjustments, we may sometimes get an object with exactly the properties (i) and (ii) that are desired. Roughly speaking, we push the difficulties off to infinity, where they disappear ...." [45]

Reformulating this heuristic argument in a more formal way yields that $\hat{\mathcal{X}}(K) = (X^*\Pi)^*(K) = \cup \{(X^*\Pi)^n(K) \mid n \geq 0\}$ [45]. When we have established results like Theorem 2 and 4(ii) (e.g. for $X = A$) then $A^* = A$ and it clearly suffices to apply a single composition $A\Pi$ on K in order to obtain $\hat{\mathcal{A}}(K)$, instead of the (possibly infinite) union $\bigcup_{n\geq 0} (A\Pi)^n$. A Canonical Form like $\hat{\mathcal{X}}(K) = X\Pi(K)$ also implies that the monoid $Mn\{\Pi,X\}$ generated by Π and X is finite (Note that the composition of transformations is associative); e.g. $Mn\{\Pi,A\}$ consists of the elements I, Π, A, ΠA and $A\Pi$.

In almost all cases to be considered in the sequel we have $XX(FIN) = X(FIN)$ and we are therefore able to establish a corresponding Canonical Form Theorem. On the other hand there exist non-pathological extensions (i.e. extensions naturally arising from well-known grammars; cf. e.g. [14]) for which $XX(FIN) \neq X(FIN)$, a Canonical Form Theorem as $\hat{\mathcal{X}}(K) = X\Pi(K)$ does not hold, and $Mn\{\Pi,X\}$ is an infinite monoid.

Apart from this introductory chapter the contents of this thesis have been divided into three parts each in turn dealing with a main aspect of generalized grammars, viz. basic properties of generalized grammars and their corresponding extensions (Part I: Chapters 2,3 and 4), extensions and their fixed points (Part II: Chapters 5 and 6), and the structure of the set of all fixed points belonging to an extension (Part III: Chapter 7). This

division is actually not so rigid, e.g. generalized context-free and regular grammars are treated in Part II (Chapter 5).

In Chapter 2 we investigate generalizations of Lindenmayer systems or L systems. Although these systems originated from developmental biology, many variations of L systems have been studied intensively in formal language theory during the last decade (cf. [31, 53, 37, 54, 52] and the references mentioned there). From the numerous different kinds of L systems defined until now, we mainly restrict our attention to the family ETOL [50] which is nowadays considered to be the central family in (context-independent) L systems theory.

An extended tabled context-independent Lindenmayer system or ETOL system consists - just as a context-free grammar - of an alphabet V, a terminal alphabet Σ, an initial symbol S, and productions of the form $\alpha \to w$ ($\alpha \in V$; $w \in V^*$). The difference between a context-free grammar and an ETOL system is however threefold: (i) An ETOL system possesses a finite number of (possibly) different sets of productions, called tables: $P_1,\ldots,P_m$ (The collection $\{P_1,\ldots,P_m\}$ is denoted by U). In a rewriting step only productions from one and the same table may be used. A context-free grammar has only one set of productions.

(ii) For an ETOL system it is required that for each table P in U and for each symbol α in V there is at least one production in P with left-hand side equal to α. So in an ETOL system there are also productions for terminal symbols, whereas a context-free grammar only contains rules for rewriting nonterminals. In an ETOL system it is therefore also allowed that $S \in V$; in a context-free grammar we have that $S \in V - \Sigma$.

(iii) The derivation relation $\Longrightarrow$ for ETOL systems differs essentially from the one for context-free grammars, viz. for each $\phi,\psi \in V^*$ the relation $\phi \Longrightarrow \psi$ holds by definition if (1) $\phi = \alpha_1\ldots\alpha_n$ with $\alpha_1,\ldots,\alpha_n \in V$;

(2) $\psi = w_1 \ldots w_n$ with $w_1, \ldots, w_n \in V^*$; and (3) for some P in U and for all i ($1 \leq i \leq n$) $\alpha_i \rightarrow w_i$ is a production in P, i.e. each symbol in ϕ is replaced (in a parallel way) according to some rule in P. (The reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$. The language generated by an ETOL system $G = (V,\Sigma,U,S)$ is defined as usual, viz. $L(G) = \{w \in \Sigma^* \mid S \Longrightarrow^* w\}$. The family of languages generated by ETOL systems is denoted by ETOL.) Thus an ETOL system is a (in general non-nested) parallel rewriting system, in the sense that at each rewriting step all symbols in a sentential form must be replaced. Rewriting according to a context-free grammar occurs in a sequential way: in rewriting a string only one nonterminal symbol is replaced. As we saw above, the resulting context-free language is equal to the one obtained by nested parallel rewriting, i.e. at each rewriting step each occurrence of a (nonterminal) symbol may be replaced or remain unaltered. Therefore we have CF $\subseteq$ ETOL [50].

The essential difference between ETOL systems and context-free grammars is (iii), whereas (i) and (ii) are only minor details caused by definitional conventions. We could define context-free grammars by also allowing productions for terminal symbols (as discussed before), an initial symbol from V, and different sets of productions, without properly enlarging the family of context-free languages (Chapter 5). Similarly, we could define ETOL systems by only allowing productions for nonterminal symbols (and requiring that $S \in V - \Sigma$), but in general it is impossible to reduce the number of tables to one [50].

Example 5. (ETOL system and ETOL language).
Consider the ETOL system $G = (V,\Sigma,U,S)$ with $V = \Sigma \cup \{S\}$, $\Sigma = \{a,b\}$, $U = \{P_1,P_2\}$, $P_1 = \{S \rightarrow SS,\ a \rightarrow a,\ b \rightarrow b\}$, $P_2 = \{S \rightarrow a,\ a \rightarrow a,\ a \rightarrow ab,\ a \rightarrow ba,\ b \rightarrow b\}$ and let $N_a(w)$ denote the number of a's occurring in w. Then $L(G) = \{w \in \Sigma^* \mid N_a(w) = 2^n$ for some $n \geq 0\}$. Note that $L(G) \in$ ETOL - CF, and hence

CF ⊂ ETOL [50]. □

In [60, 57, 49, 50] it has been observed that - similar to the context-free case - each table in an ETOL system G is just a finite substitution over the alphabet of G. Consequently, the set of sentential forms generated by G is obtained by iterating (non-nested) finite substitutions on the initial symbol of G.

Example 6. (tables interpreted as finite substitutions).
The ETOL system G of Example 5 can be written as (V,Σ,U_0,S) where $U_0 = \{\tau_1,\tau_2\}$. The two (non-nested) finite substitutions τ_1 and τ_2 over V are defined by $\tau_1(S) = \{SS\}$; $\tau_1(a) = \{a\}$; $\tau_1(b) = \{b\}$; $\tau_2(S) = \{a\}$; $\tau_2(a) = \{a,ab,ba\}$: $\tau_2(b) = \{b\}$. Clearly, $L(G) = U_0^*(S) \cap \Sigma^*$, where $U_0^*(S)$ is defined by $U_0^*(S) = \cup \{\sigma_n \ldots \sigma_1(S) \mid n \geq 0;\ \sigma_i \in U_0,\ 1 \leq i \leq n\}$. □

Replacing the finite substitutions of an ETOL system G by K-substitutions (for an arbitrary family K) over the alphabet of G, as we showed for context-free grammars, yields the notion of K-iteration grammar, which was originally introduced and investigated by Van Leeuwen [60] and Salomaa [57].

So a K-iteration grammar G (cf. Chapter 2, Section 1) consists of an alphabet V, a terminal alphabet Σ ($\Sigma \subseteq V$), an initial symbol $S \in V$, a finite set I (the index alphabet of G) and a finite set $U = \{\tau_i \mid i \in I\}$ of K-substitutions over V satisfying $\tau_i(\alpha) \neq \emptyset$ ($\alpha \in V$; $i \in I$). The language generated by G is $L(G) = U^*(S) \cap \Sigma^*$ where $U^*(S)$ is defined as in Example 6. A K-iteration grammar may be viewed as an ETOL system in which each table τ contains a countable number of productions, viz. $\alpha \to w$ iff $w \in \tau(\alpha)$, whereas these countably many alternatives for each symbol are restricted by the family K. The family of languages generated by K-iteration grammars is called the hyper-algebraic extension H(K) of K. (Clearly, H(FIN) = ETOL).

In Chapter 2 we investigate K-iteration grammars and hyper-algebraic extensions. In particular we establish the hyper-algebraic variants of

Theorems 1, 2, 3 and 4: instead of full A-AFL's (super-AFL's) we obtain full H-AFL's (hyper-AFL's, i.e. full AFL's closed under iterated (non-nested) substitution [60, 57, 9]) whereas CF, A and $\hat{\mathcal{A}}$ are replaced by ETOL, H and $\hat{\mathcal{H}}$ respectively ($\hat{\mathcal{H}}(K)$ is the smallest full H-AFL, i.e. the smallest family closed under Π and H, including K). The basic fact that HH(FIN) = H(FIN), i.e. that ETOL is closed under iterated substitution was shown in [9].

In proving these theorems and other properties of K-iteration grammars the concept of control language is very useful (For control mechanisms on context-free grammars, we refer to [56] for a survey). The idea is that we only allow those derivations according to a K-iteration grammar, of which the sequences of substitution indices $i_1 \ldots i_n$ belong to a certain given control language over I. This control language is in turn taken from a given family of control languages Γ. The Γ-controlled hyper-algebraic extension $H(\Gamma,K)$ of K consists of all languages generated by K-iteration grammars with control languages from Γ. Obviously, controlled K-iteration grammars are a generalization of controlled ETOL systems [43, 21, 46] in the same way as K-iteration grammars are of ETOL systems.

Example 7. (controlled iteration grammar).
Consider the REG-controlled REG-iteration grammar $G = (V,\Sigma,U,I,M,S)$ where $V = \Sigma = \{a,b\}$, $S = a$, $I = \{\alpha,\beta\}$, M is the control language $\{\alpha^n\beta \mid n \geq 0\}$, and U contains the substitutions $\tau_\alpha(a) = \{a^2\}$; $\tau_\alpha(b) = \{b\}$; $\tau_\beta(a) = b^*ab^*$; $\tau_\beta(b) = \{b\}$. Then $L(G) = M(S) \cap \Sigma^* = \cup \{\tau_\beta\tau_\alpha^n(S) \mid n \geq 0\} \cap \Sigma^* = \{w \in \Sigma^* \mid N_a(w) = 2^n \text{ for some } n \geq 0\} \in H(REG,REG)$. □

A family Γ is closed under full marking if for all $M \subseteq I^*$ in Γ and for all symbols a and b not in I, the language aMb is in Γ.

The main result concerning Γ-controlled K-iteration grammars now reads as follows: if K is a prequasoid, and if Γ is closed under full marking, union or concatenation, and Kleene *, then $H(\Gamma,K)$ is a full hyper-AFL

including K, Γ, H(K) and ETOL. Together with the fact that control by regular languages may freely be used, i.e. H(REG,K) = H(K), this enables us to establish the desired Canonical Form for full hyper-AFL's: $\hat{\mathcal{H}}(K) = H\Pi(K)$. Moreover, we prove in Chapter 2 some basic properties of controlled iteration grammars (assuming a few weak conditions on Γ and K), such as (i) for each (Γ-controlled) K-iteration grammar the number of substitutions can be reduced to two, (ii) if $K \subseteq RE$ (where RE is the family of recursively enumerable languages), and if each language in RE is the homomorphic image of a language in Γ (e.g. Γ is the family of context-sensitive languages), then $H(\Gamma,K) = RE$, and (iii) if the emptiness problem is decidable in K [and in Γ], then both the emptiness and the membership problem are decidable in H(K) [and $H(\Gamma,K)$]. Most proof techniques in Chapter 2 are generalizations of known arguments for the finite case, i.e. for ETOL [50, 9].

Chapter 3 is devoted to generalizations of deterministic ETOL systems. A deterministic ETOL system or EDTOL system [50] is an ETOL system $G = (V,\Sigma,U,S)$ satisfying the condition that for each P in U and for each α in V there is exactly one production $\alpha \to w$ in P for some $w \in V^*$. So each table may be viewed as a ONE-substitution or homomorphism.

Example 8. (EDTOL system and language).
Consider the EDTOL system $G = (V,\Sigma,U,S)$ where $V = \Sigma \cup \{S,A\}$, $\Sigma = \{a,b\}$, $U = \{P_1,P_2\}$ with $P_1 = \{S \to A^3, A \to aA, a \to a, b \to \mathbf{b}\}$ and $P_2 = \{S \to S, A \to b, a \to a, b \to \mathbf{b}\}$. Then $L(G) = \{a^n b a^n b a^n b \mid n \geq 0\}$. □

Since we want to generalize the notion of determinism, and as replacing each ONE-substitution in an EDTOL system by a K-substitution again yields K-iteration grammars - which are nondeterministic in nature - we follow another approach. Typical for applying a table of an EDTOL system on a sentential form v is that each occurrence of a symbol α in v is replaced by the same word w (assuming that $\alpha \to w$ is a production). This is the essential

idea behind the following concept of deterministically substituting K-languages into words and languages: a deterministic K-substitution or dK-substitution τ over V is a mapping $\tau : V \to K$ such that $\tau(\alpha) \subseteq V^*$ for all α in V, as usual. But this mapping 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 α in $V\}$ and to languages over V by $\tau(L) = \cup \{\tau(w) \mid w \in L\}$ for each $L \subseteq V^*$.

A deterministic substitution τ, viewed as rewriting mechanism, is context-independent in the sense that the rewriting of two different symbols is independent (but not two different occurrences of the same symbol as in the nondeterministic case!).

Example 9. (dREG-substitution).

Let $\tau : \{a,b\} \to$ REG be a dREG-substitution over $\{a,b\}$ with $\tau(a) = a^*$ and $\tau(b) = \{b,b^2\}$. Then $\tau(abab) = \{a^m u a^m u \mid m \geq 0;\ u \in \{b,b^2\}\}$. If we interpret τ as an ordinary K-substitution, then $\tau(abab) = \{a^m u a^n v \mid m,n \geq 0;$ $u,v \in \{b,b^2\}\}$. □

Sometimes we call an ordinary K-substitution a nondeterministic K-substitution or nK-substitution in order to avoid confusion; however, "K-substitution" always means nK-substitution.

If we interpret in the definition of [Γ-controlled] K-iteration grammar each K-substitution as a dK-substitution we obtain the notion of [Γ-controlled] dK-iteration grammar; the corresponding language family is called the [Γ-controlled] dhyper-algebraic extension $\eta(K)$ [$\eta(\Gamma,K)$] of K. It is clear that $\eta(ONE) = EDTOL$, but it is also true that $\eta(FIN) = EDTOL$.

Example 10. (dFIN-iteration grammar).

Consider the dFIN-iteration grammar $G = (V,\Sigma,U,I,S)$ where $V = \Sigma \cup \{S,A\}$, $\Sigma = \{a,b\}$, $I = \{i\}$, $U = \{\tau_i\}$ with $\tau_i(S) = \{A^3\}$, $\tau_i(A) = \{aA,b\}$, $\tau_i(a) = \{a\}$, $\tau_i(b) = \{b\}$. Then $L(G) = \{a^n b a^n b a^n b \mid n \geq 0\} \in \eta(FIN)$. □

For $\eta(K)$ and $\eta(\Gamma,K)$ we show in Chapter 3 results analogous to Theorems 1, 2, 3 and 4 together with some basic properties similar to those obtained for $H(K)$ and $H(\Gamma,K)$. Obviously, due to the different closure properties of EDTOL, we now need basically different sets of operations, viz. instead of prequasoids, full AFL's and full hyper-AFL's we use in the deterministic case ad hoc structures such as pseudoids (i.e. families including SYMBOL and closed under homomorphism and intersection with regular sets), full Quasi Abstract Families of Languages or full QAFL's (i.e. pseudoids closed under union, concatenation and Kleene *; note that a full AFL is a full QAFL closed under inverse homomorphism), and full dhyper-QAFL's (i.e. full QAFL's closed under iterated deterministic substitution). In general the proofs for the deterministic case are more involved and we need additional conditions on Γ.

In Chapter 4 we investigate generalizations of linear macro grammars. Macro grammars have been introduced in [17] in order to model certain context-dependent features (e.g. declaration of variables) in the syntax of programming languages. They may however be viewed as context-independent rewriting systems which have trees (or terms) as sentential forms [47].

A linear macro grammar (or linear basic grammar) consists of a ranked alphabet Φ (i.e. with each element of Φ a unique nonnegative integer - its rank or its number of arguments - is associated), a terminal alphabet Σ, an initial symbol $S \in \Phi$ of rank 0, i.e. without arguments, and a finite set P of productions of the form

$$A(x_1,\ldots,x_n) \rightarrow B(w_1,\ldots,w_m)$$
$$A(x_1,\ldots,x_n) \rightarrow w_0 \qquad (*)$$

where $A,B \in \Phi$ (having rank n and m respectively), $x_1,\ldots,x_n$ $(n \geq 0)$ are formal arguments, and $w_0,w_1,\ldots,w_m \in (\Sigma \cup \{x_1,\ldots,x_n\})^*$. The derivation relation is defined in the following way: application of the rules in $(*)$ to a sentential form $A(u_1,\ldots,u_n)$ with $u_i \in \Sigma^*$ yields $B(h(w_1),\ldots,h(w_m))$ and

$h(w_0)$ respectively, where h is the homomorphism defined by $h(x_i) = u_i$ $(1 \leq i \leq n)$ and $h(\alpha) = \alpha$ for $\alpha \in \Sigma$, i.e. the formal arguments x_i are replaced by the actual arguments u_i. The linear macro (or linear basic) language generated by such a grammar consists of all words over Σ derivable from S.

Example 11. (linear macro grammar).

Consider $G = (\Phi,\Sigma,P,S)$ with $\Phi = \{S,A\}$ (S has rank 0; A has rank 1), $\Sigma = \{a,b\}$, and P contains the rules $S \to A(b)$, $A(x) \to A(ax)$, $A(x) \to x^3$. Then $L(G) = \{a^n b a^n b a^n b \mid n \geq 0\}$. □

Several particular cases of generalized linear macro grammars have been investigated: in [11] the words $w_0,\ldots,w_m$ in (*) are replaced by finite languages over $\Sigma \cup \{x_1,\ldots,x_n\}$ (the corresponding language family is denoted by ELB), whereas in [16] regular languages over $\Sigma \cup \{x_1,\ldots,x_n\}$ are used in a similar way.

In Chapter 4 we consider the general case of replacing $w_0,\ldots,w_m$ in (*) by languages from an arbitrary family K, resulting in rules of the form

$$A(x_1,\ldots,x_n) \to B(L_1,\ldots,L_m)$$
$$A(x_1,\ldots,x_n) \to L_0 \qquad (**)$$

where $L_0,\ldots,L_m$ are languages from K (over the alphabet $\Sigma \cup \{x_1,\ldots,x_n\}$). We consider two different modes of derivation for these generalized grammars: the OI (outside-in) and the IO (inside-out) modes, inspired by the analogous concepts for arbitrary macro grammars [17].

In the OI mode of derivation application of (**) to a sentential form $A(U_1,\ldots,U_n)$ with $U_i \subseteq \Sigma^*$ yields $B(\tau(L_1),\ldots,\tau(L_m))$ and $\tau(L_0)$ respectively, where τ is the substitution defined by $\tau(x_i) = U_i$ $(1 \leq i \leq n)$ and $\tau(\alpha) = \{\alpha\}$ for $\alpha \in \Sigma$. Note that τ is a substitution in languages of K. The language OI-generated by the grammar consists of the union of all languages over Σ derivable from S. This language is therefore obtained by iterated substitution in languages of K.

In the IO mode of derivation application of (**) to a sentential form $A(u_1,\ldots,u_n)$ with $u_i \in \Sigma^*$ yields $B(v_1,\ldots,v_m)$ with $v_i \in h(L_i)$, where h is the homomorphism defined as before in the non-generalized case.

In this way we obtain the notions of OI-extended and IO-extended linear basic K-grammars. The corresponding language families are denoted by $LB_{OI}(K)$ and $LB_{IO}(K)$.

Example 12. (OI- and IO-extended linear basic FIN-grammar). Consider $G = (\Phi,\Sigma,P,S)$ with $\Phi = \{A,S\}$ (S has rank 0, A has rank 1), $\Sigma = \{a,b\}$, and P consists of the rules $S \to A(\{a,b\})$, $A(x) \to A(xx)$, $A(x) \to x$ (As usual, we write xx for $\{xx\}$). An example of an OI-derivation [IO-derivation] is: $S \underset{OI}{\Longrightarrow} A(\{a,b\}) \underset{OI}{\Longrightarrow} A(\{a,b\}^2) \underset{OI}{\Longrightarrow} A(\{a,b\}^4) \underset{OI}{\Longrightarrow} \{a,b\}^4$ [$S \underset{IO}{\Longrightarrow} A(b) \underset{IO}{\Longrightarrow} A(b^2) \underset{IO}{\Longrightarrow} A(b^4) \underset{IO}{\Longrightarrow} b^4$]. It is easy to show that the corresponding languages - denoted by $L_{OI}(G)$ and $L_{IO}(G)$ - are $L_{OI}(G) = \{w \in \Sigma^* \mid$ the length of w is 2^n for some $n \geq 0\}$ and $L_{IO}(G) = \{a^{2^n} \mid n \geq 0\} \cup \{b^{2^n} \mid n \geq 0\}$. □

The first main result of Chapter 4 establishes the equalities $H(K) = LB_{OI}(K)$ and $\eta(K) = LB_{IO}(K)$ under weak assumptions on K. These are the essential generalizations of the corresponding equalities for the finite case, viz. ETOL = ELB and EDTOL = LB (where LB is the family generated by ordinary linear basic grammars) which were both proved in [11]. This shows that iterated substitution is the operation corresponding to generalized linear macro grammars, as could be imagined from the way derivations in these grammars are defined. Hence the operation of iterated substitution has two different but equivalent grammatical characterizations: by K-iteration grammars and by linear basic K-grammars. This result implies that the analogues of Theorems 1 - 4 for LB_{OI} and LB_{IO} follow from those for H and η established previously.

A similar equivalence also holds for Γ-controlled [d]K-iteration grammars and Γ-controlled OI [IO]-extended linear basic K-grammars.

Corresponding control words however must be reversed; this reflects the point of view that linear basic K-grammars and iteration grammars compute each others fixed points (of the set of equations derived from the grammar; cf. [11, 15, 30]).

The second main result of Chapter 4 relates (controlled) extended linear basic K-grammars to a class of nondeterministic program schemes without tests, called register programs. (For the relevance of the theory of program schemes in computer science and its relation to formal languages we refer to [6, 13, 27]). Such a register program may be viewed as an IO- or OI-extended linear basic K-grammar of a special type. The IO-version of this grammar corresponds to the use of nonrecursive function procedures in the register program, whereas the OI-variant corresponds to register programs which compute on sets.

Example 13. (register program).

Consider the register program P (with a single register x)

start(x:={a,b}); repeat(x:=xx); halt(x)

where the repeat-statement has to be executed any number of times. In the OI-interpretation the register x contains a language, whereas in the IO-interpretation it contains a word and {a,b} may be viewed as a (nondeterministic) procedure which returns either a or b. As an example consider the following computations represented by the successive contents of x:

OI: $\{a,b\}$, $\{a,b\}^2$, $\{a,b\}^4$

IO: b, b^2, b^4

Under these interpretations P computes (or generates) the languages $LB_{OI}(G)$ and $LB_{IO}(G)$ of Example 12. □

The reader is referred to Chapter 4 for a more complete survey of the relation between register programs, (controlled) extended linear basic K-grammars, and (controlled) K-iteration grammars.

Chapter 5 is mainly devoted to context-free

K-grammars and regular K-grammars and the corresponding transformations on language families, viz. the algebraic extension A(K) and the rational extension R(K) of K. A regular K-grammar is defined (just as an ordinary regular grammar can be characterized; cf. [56] II Theorem 5.5) as a not self-embedding context-free K-grammar, i.e. a context-free K-grammar $G = (V,\Sigma,\tau,S)$ with the property that for all $\alpha \in V$, and for all $n \geq 1$ the following implication holds: if $w_1\alpha w_2 \in \tau^n(\alpha)$, then either $w_1 = \lambda$ or $w_2 = \lambda$ $(w_1,w_2 \in V^*)$.

In Chapter 5 we prove Theorems 1 - 4 and the analogous results for the algebraic and rational extension. In the latter case the corresponding operation is substitution (not iterated due to the not self-embedding restriction), and thus regular K-grammars provide a grammatical characterization of full substitution-closed AFL's.

Actually we define context-free and regular K-grammars in such a way that they possess a finite set of nested K-substitutions instead of a single substitution (analogous to the definition of iteration grammar). Under simple conditions on K this finite number of substitutions can be reduced to a single substitution (contrary to the case of iteration grammars where in general only a reduction to two substitutions is possible). Therefore a control mechanism on the order of substitutions (as for iteration grammars) is uninteresting in the algebraic and rational case.

Finally, we consider in Chapter 5 a few further particularizations of the rational extension providing grammatical characterizations (in the sense of Theorems 1 - 4) of full AFL's, full semi-AFL's and related structures. These results are similar to those obtained in [35, 44].

Before we turn to the contents of Chapters 6 and 7 we summarize the main language families discussed so far in a diagram (Fig. 1.1); for the correctness of this diagram we refer to [32, 56, 50, 12]. The chain $REG \subset CF \subset CS \subset RE$, where CS is the family of context-sensitive languages, is usually referred to as the Chomsky hierarchy.

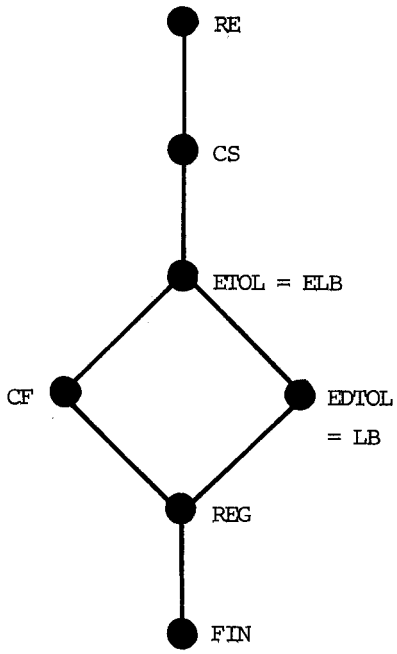


Figure 1.1

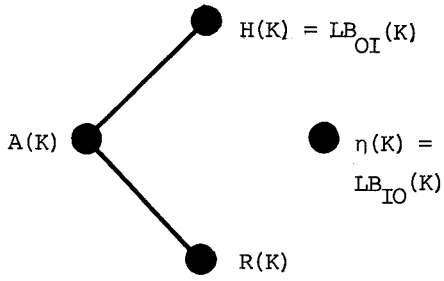


Figure 1.2

The extensions corresponding to the "square" in Fig. 1.1 are mentioned in Fig. 1.2. Note that in Fig. 1.1 a line between families means proper inclusion, whereas in Fig. 1.2 it only denotes inclusion ($R(K) \subseteq A(K) \subseteq H(K)$, for all K, follows from the definitions of R, A and H). Whether these inclusions are indeed proper depends of course on K. E.g. if K = FIN, then Fig. 1.2 coincides with the square of Fig. 1.1; on the other hand if K equals RE or the empty family, then Fig. 1.2 collapses into a single point. In general η is incomparable with H, A and R (see Chapter 3). This is not very surprising since η is based on a different notion of substitution, viz. dK-substitution, than the other extensions, which are based on nK-substitutions (H) or restricted versions of them (A and R). If K equals ONE, then the concepts of dK- and nK-substitution coincide: in both cases we obtain homomorphisms, of which dK-substitutions and nK-substitutions are (different) generalizations.

In Chapter 6 we consider monoids of transformations (or operators) on language families, finitely generated by closure operators such as A, Π and $\hat{\mathcal{A}}$. (The monoid operation is the composition of operators, and the unit of the monoid is the identity operator I(K) = K for all K). These monoids are provided with a partial order induced by set-theoretical inclusion of language families, viz. for operators Γ_1 and Γ_2 we have by definition $\Gamma_1 \leq \Gamma_2$ if $\Gamma_1(K) \subseteq \Gamma_2(K)$ for all K. In all cases to be considered in Chapter 6 these monoids are finite (due to Theorem 4), and for a few of them we determine their exact structure, i.e. their partial order and their multiplication table.

The structure of these monoids provides additional insight in (in)dependency of operations on languages (cf. [20]) and in the effect of successively applying language-theoretic closure operators.

In Chapter 7 we investigate the structure (with respect to set-theoretical inclusion) of the class of all full A-AFL's (full H-AFL's, etc.), i.e. of the set consisting of all fixed points of the transformation $\hat{\mathcal{A}}$ ($\hat{\mathcal{R}}$, etc.). Since each full A-AFL K may be viewed as an algebra, i.e. a carrier set K closed under a fixed collection of finitary operations on K (viz. the operations defining a full A-AFL), results established in universal algebra become applicable in AFL-theory. In particular we prove certain (lattice-theoretic) characterization theorems on incomparable elements in the algebraic (or compactly generated) lattice of subalgebras of a given algebra, inspired by results on incomparable full AFL's due to Ginsburg and Spanier [24]. Using lattice theory and universal algebra we are able to obtain in a uniform way Ginsburg and Spanier like results for full X-AFL-structures considered (such as X = R, A, H). This shows how little of the full AFL (or full A-AFL) properties one actually needs in order to obtain such characterization theorems.

It will be clear that a major part of this thesis may be considered as (i) a straightforward contribution to the theory of generalized grammars (Chapters 2 - 5), as initiated in [60, 57, 61] and developed further in [65, 51, 55, 62, 64, 66, 14, 63], and (ii) a unifying approach to full AFL-structures related to these generalized grammars (Chapters 5 - 7). Some ideas seem to be (more or less) original. On these aspects we now focus our attention.

- Deterministic substitution has been considered previously in the literature, in particular in the finite case (dFIN-substitution; cf. [10, 34, 48, 59]) or in relation with substituting trees in trees [15, 48]. However in Chapter 3 it is investigated systematically from the point of view of generalizing a certain part of L systems theory (viz. EDTOL systems) to the more algebraic framework of dK-iteration grammar, similar to the nondeterministic case (ETOL systems versus K-iteration grammars).
- The notions of (controlled) OI-extended and IO-extended linear basic K-grammar as well as their relation to register programs (Chapter 4) are new. The fact that (controlled) OI-extended linear basic K-grammars are equivalent to (controlled) K-iteration grammars is not very surprising (cf. [11, 16, 30, 15]) but the equivalence of (controlled) IO-extended linear basic K-grammars and (controlled) dK-iteration grammars is less obvious.
- The concepts of regular K-grammar (or not self-embedding context-free K-grammar) and rational extension appear in Chapter 5 for the first time, as well as the grammatical characterization of full substitution-closed AFL's as full R-AFL's (i.e. families which are both a prequasoid and a fixed point of the rational extension R).
- Although the general idea of providing grammars (rewriting systems) with a control mechanism is by no means original [56, 21, 43, 46], throughout this thesis (Chapters 2 - 4) control is used as a tool to facilitate proofs. In this way we obtain not only more general results (viz. for controlled

grammars) but are also able to cover the uncontrolled cases, since for all generalized grammars considered the use of regular control does not increase the generating power of the uncontrolled devices. We emphasize that the main results for the uncontrolled grammars may be established in a direct way (i.e. without using the controlled variants) but in general the proofs become less transparent.

- In Chapter 7 we provide a (first) attempt to determine to what extent several results established for full AFL's are actually lattice-theoretic or universally algebraic in nature.

In the previous part of this chapter our motivation to study generalized grammars and related extensions has been given in a rather implicit way. Therefore we now summarize more explicitly our main reasons to investigate these grammars, the corresponding extensions and their fixed points.

(1) The idea of generalized grammar provides a uniform framework for investigating iterated context-independent rewriting. So dealing with (types of) substitutions instead of particular kinds of productions enables us to obtain more (mathematical) insight in both the similarities and differences between several classes of context-independent grammars [60, 57].

(2) In establishing properties of generalized grammars we have to make assumptions on the substitutions involved in these grammars or, equivalently, we have to put restrictions on the family K. Since most properties of generalized grammars considered until now [14, 51, 55, 57, 60 - 66] are generalizations of results already obtained for the original (ordinary) grammars - i.e. the case that K equals FIN - these restrictions on K make clear on which properties of the finite languages the original results for ordinary grammars actually depend. In other words [60, 61, 63], in the theory of generalized grammars we are looking for structural proofs

rather than combinatorial arguments usually applied in the finite case, i.e. in dealing with ordinary grammars.

(3) We already pointed out that each generalized grammar may be viewed as an operation on languages and that as a consequence there exists a close correspondence between fixed points of extensions and closure properties of language families. So well-known sets of closure properties as abstracted in several full AFL-structures (such as full substitution-closed AFL, full super-AFL, full hyper-AFL) can be characterized in terms of fixed points of extensions. In this way we obtain a formal grammatical characterization of these full AFL-structures (Chapters 5 and 6).

(4) The well-known connection between language families and classes of program schemes [6, 13, 27] also exists between language families generated by generalized grammars (viz. extended linear basic K-grammars, iteration grammars) and certain classes of nondeterministic polyadic program schemes (viz. register programs, Chapter 4).

(5) Since each full X-AFL (X = R, A, etc.) may be viewed as an algebra and particularly as a subalgebra of the family of all languages over Σ_ω, universal algebra and lattice theory can be used to re-examine results in AFL-theory. In this way we obtain better insight in the question whether certain properties of full AFL-structures (such as those established in [24]) are actually algebraic rather than language-theoretic in character (Chapter 7).

We remark that (1) and (2) are rather general motives to study generalized grammars and extensions [57, 60, 61, 63], whereas (3) - (5) (in particular (3)) apply more specifically to the contents of this thesis.

We conclude this chapter with a few suggestions concerning the directions in which this research could be continued.

First of all we have the general problem of generalizing an interesting

result for ordinary grammars to a corresponding theorem for generalized grammars. We emphasize that this problem is by no means restricted to closure properties (to which this thesis is mainly devoted). Several results in this direction have already been established [51, 61, 63 - 65] and it is very likely that other interesting examples could be found. In particular one may consider generalized macro grammars which are not necessarily linear as in Chapter 4.

Secondly, deterministic iteration grammars deserve further study. In particular we do not know the properties of (i) dK-iteration grammars with a single dK-substitution (The nondeterministic variant has been considered in [14]), (ii) dK-iteration grammars with nested dK-substitutions, and (iii) nested dK-iteration grammars satisfying the not self-embedding property.

Thirdly, some problems remain in the theory of full hyper-AFL's (similar to those already considered for full AFL's): e.g. (i) do there exist extensive and monotonic operators which, when iteratively applied to a full H-AFL, give rise to an infinite hierarchy of full H-AFL's? (cf. [26]), (ii) does there exist an operator X on language families such that for each full H-AFL K, X(K) = K implies that K is H-coprime, i.e. for all full H-AFL's $K_1,\ldots,K_n$ $K \subseteq H(K_1 \cup \ldots \cup K_n)$ implies $K \subseteq K_i$ for some i (The existence of such an operator X could be a main tool in finding largest incomparable full H-AFL's; cf. Chapter 7).

Finally, the research started in Chapter 7 yields the general question whether certain other propositions obtained in AFL-theory (cf. e.g. [19, 26]) also have corresponding generalizations in universal algebra or in lattice theory.

References

1. A.V. Aho, J.D. Ullman,"The Theory of Parsing Translation and Compiling", Vol. I (1972), Vol. II (1973), Prentice Hall, Englewood Cliffs, N.J.

2. J.W. Backus, et al., Revised Report on the Algorithmic Language ALGOL 60,

(1964) A/S Regnecentralen, Copenhagen.

3. Y. Bar-Hillel,"Language and Information", (1964) Addison-Wesley, Reading, Mass.

4. Y. Bar-Hillel, M. Perles, E. Shamir, On formal properties of simple phrase structure grammars, Zeitschr. für Phonetik, Sprachwiss. und Kommunikationsforschung 14 (1961) 143-172 (Also Chapter 9 in [3]).

5. H. Brandt Corstius,"Algebraïsche Taalkunde", (1974) Oosthoek, Utrecht.

6. A.K. Chandra, On the Properties and Applications of Program Schemas, (1973) Ph.D.Thesis Stanford University, Stanford, Ca.

7. N. Chomsky, Formal Properties of Grammars, pp. 323-418 in [39].

8. N. Chomsky, G.A. Miller, Introduction to the Formal Analysis of Natural Languages, pp. 269-321 in [39].

9. P.A. Christensen, Hyper-AFL's and ETOL Systems, DAIMI PB-35 (1974), University of Aarhus, Denmark (cf. pp.254-257 in [53]).

10. J. Dassow, Über quasideterministische Sprachen, Rostocker Mathematisches Kolloquium 1 (1976) 51-60.

11. P.J. Downey, Formal Languages and Recursion Schemes, (1974) Ph.D.Thesis Harvard University, Cambridge, Mass.

12. A. Ehrenfeucht, G. Rozenberg, On some context-free languages that are not deterministic ETOL languages, Rev. Française Automat. Informat. Rech. Opérat. IT 11 (1977) 273-291.

13. J. Engelfriet,"Simple Program Schemes and Formal Languages", (1974) Lecture Notes in Computer Science Vol. 20, Springer, Berlin-Heidelberg-New York.

14. J. Engelfriet, Iterating iterated substitution, Theor. Comp. Sci. 5 (1977) 85-100.

15. J. Engelfriet, E. Meineche Schmidt, IO and OI, Part I J. Comp. System Sci. 15 (1977) 328-353, Part II J. Comp. System Sci. 16 (1978) 67-99.

16. J. Engelfriet, E. Meineche Schmidt, J. van Leeuwen, Stack machines and classes of nonnested macro-languages (1977) Report RUU-CS-77-2, University

of Utrecht.

17. M.J. Fischer, Grammars with MACRO-like productions, (1968) Ph.D.Thesis Harvard University, Cambridge, Mass.(cf. pp. 131-142 in Proc. 9th Ann. Symp. on Switching and Automata Theory, 1968).

18. S. Ginsburg,"The Mathematical Theory of Context-Free Languages", (1966) McGraw-Hill, New York.

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

20. S. Ginsburg, S.A. Greibach, J.E. Hopcroft, Studies in Abstract Families of Languages, (1969) Mem. Amer. Math. Soc. 87.

21. S. Ginsburg, G. Rozenberg, TOL schemes and control sets, Inform. Contr. 27 (1975) 109-125.

22. S. Ginsburg, E.H. Spanier, Substitution in families of languages, Inform. Sci. 2 (1970) 83-110.

23. S. Ginsburg, E.H. Spanier, AFL with the semilinear property, J. Comp. System Sci. 5 (1971) 365-396.

24. S. Ginsburg, E.H. Spanier, On incomparable Abstract Families of Languages (AFL), J. Comp. System Sci. 9 (1974) 88-108.

25. S.A. Greibach, Full AFL's and nested iterated substitution, Inform. Contr. 16 (1970) 7-35.

26. S.A. Greibach, Syntactic operators on full semi-AFL's, J. Comp. System Sci. 6 (1972) 30-76.

27. S.A. Greibach,"Theory of Program Structures: Schemes, Semantics, Verification", (1975) Lecture Notes in Computer Science Vol. 36, Springer, Berlin-Heidelberg-New York.

28. J. Gruska, A characterization of context-free languages, J. Comp. System Sci. 5 (1971) 353-364.

29. J. Gruska, Generalized context-free grammars, Acta Cybernetica 2 (1973) 35-37.

30. G.T. Herman, A biologically motivated extension of ALGOL-like languages, Inform. Contr. 22 (1973) 487-502.

31. G.T. Herman, G. Rozenberg,"Developmental Systems and Languages", (1975) North-Holland, Amsterdam.

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

33. J. Král, A modification of a substitution theorem and some necessary and sufficient conditions for sets to be context-free, Math. Syst. Theory 4 (1970) 129-139.

34. M. Kudlek, Comparing several ways of context-independent parallel rewriting, pp. 122-130 in D. Siefkes (Ed.):"GI-4. Jahrestagung", Lecture Notes in Computer Science Vol. 26 (1975) Springer, Berlin-Heidelberg-New York.

35. D.J. Lewis, Closure of families of languages under substitution operators, pp. 100-108 in Proc. 2nd Ann. A.C.M. Symp. Theory of Computing (1970).

36. A. Lindenmayer,"Developmental Systems and Languages in Their Biological Context", (Chapter 0 of [31]).

37. A. Lindenmayer, G. Rozenberg (Eds.),"Automata, Languages, Development", (1976) North-Holland, Amsterdam.

38. P. Lorenzen,"Einführung in die operative Logik und Mathematik", Zweite Auflage (1969), Springer, Berlin-Heidelberg-New York.

39. R.D. Luce, R.B. Bush, E. Galanter (Eds.),"Handbook of Mathematical Psychology", (1963) Vol. II, Wiley, New York.

40. A.A. Markov,"The Theory of Algorithms", Akademiya Nauk SSSR (1954), Israel Program for Scientific Translations (1962) Jerusalem.

41. H.A. Maurer,"Theoretische Grundlagen der Programmiersprachen", (1969) Bibliographisches Institut, Mannheim.

42. I.P. McWirther, Substitution expressions, J. Comp. System Sci. 5 (1971) 629-637.

43. M. Nielsen, EOL systems with control devices, Acta Informatica 4 (1975) 373-386.

44. M. Nivat, Opérateurs sur les familles de langages, (1975) I.R.I.A. Rapport de recherche, Le Chesnay, France.

45. R.S. Pierce, Closure spaces with applications to ring theory, pp. 565-616 in "Lectures on Rings and Modules", Lecture Notes in Mathematics Vol. 246 (1972), Springer, Berlin-Heidelberg-New York.

46. P.J.A. Reusch, Regulierte TOL-Systeme I, Bericht Nr. 81, (1974), G.M.D., Bonn, FGR.

47. W.C. Rounds, Mappings and grammars on trees, Math. Syst. Theory. 4 (1970) 257-287.

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

49. G. Rozenberg, TOL systems and languages, Inform. Contr. 23 (1973) 357-381.

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

51. G. Rozenberg, On slicing of K-iteration grammars, Inform. Process. Letters 4 (1976) 127-131.

52. G. Rozenberg, M. Penttonen, A. Salomaa, Bibliography of L systems, Theor. Comp. Sci. 5 (1977) 339-354.

53. G. Rozenberg, A. Salomaa (Eds.),"L Systems", (1974) Lecture Notes in Computer Science Vol. 15, Springer, Berlin-Heidelberg-New York.

54. G. Rozenberg, A. Salomaa, The Mathematical Theory of L Systems, pp. 161-206 in J.T. Tou,"Advances in Information Systems Science", Vol. 6 (1976) Plenum Press, New York.

55. G. Rozenberg, D. Wood, A note on K-iteration grammars, Inform. Process. Letters 4 (1976) 162-164.

56. A. Salomaa,"Formal Languages", (1973) Academic Press, New York.

57. A. Salomaa, Macros, iterated substitution and Lindenmayer AFL's, DAIMI PB-18 (1973), University of Aarhus, Denmark (cf. pp. 250-253 in [53]).

58. N.A. Sanin,"Constructive Real Numbers and Constructive Function Spaces", Translation of Math. Monographs Vol. 21 (1968) Amer. Math. Soc., Providence R.I.

59. R. Siromoney, G. Siromoney, Parallel OL-languages, Internat. J. Comp. Math. 5A (1975) 109-123.

60. J. van Leeuwen, F-iteration languages, Memorandum (1973), University of California, Berkeley.

61. J. van Leeuwen, A generalization of Parikh's theorem in formal language theory, pp. 17-26 in J. Loeckx (Ed.),"Automata, Languages and Programming, 2nd Colloquium"(1974) Lecture Notes in Computer Science Vol. 14, Springer, Berlin-Heidelberg-New York.

62. J. van Leeuwen, A study of complexity in hyper-algebraic families, pp. 323-333 in [37].

63. J. van Leeuwen, Effective constructions in well-partially-ordered free monoids, Discrete Math. (to appear).

64. J. van Leeuwen, D. Wood, A decomposition theorem for hyper-algebraic extensions of language families, Theor. Comp. Sci. 1 (1976) 199-214.

65. D. Wood, A note on Lindenmayer systems, Szilard languages, spectra, and equivalence, Internat. J. Comp. Inform. Sci. 4 (1975) 53-62.

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

P A R T I

TRANSFORMATIONS INDUCED BY GENERALIZED GRAMMARS

Chapter 2

Controlled Iteration Grammars and Full Hyper-AFL's

Reprinted from *Information and Control 34 (1977) 248 - 269*

Remark

In this chapter a family is required to be closed under isomorphism ("renaming of symbols").

The [Γ-controlled] hyper-algebraic extension is denoted by (K)ITER [(Γ,K)ITER] instead of H(K) [H(Γ,K)]. Similarly, $(K)ITER^{(m)}$ and $(\Gamma,K)ITER^{(m)}$ are used for the m-restricted variants.

Note, that according to the definition of K-iteration grammar each language $\tau_i(\alpha)$ is nonempty. This condition should however be dropped in Section 7.

Errata

p. 42 (252) line 22: change "that" to "than".

p. 43 (253) line 15: change "*(i≤j≤n)*" to "*(1≤j≤n)*".

p. 49 (259) line 17: change "*, *Then*" to "*. *Then*".

p. 42 (252), 50 (260), 58 (268) running head:
change "A." to "R.".

Reprinted from INFORMATION AND CONTROL
All Rights Reserved by Academic Press, New York and London

Vol. 34, No. 3, July 1977
Printed in Belgium

Controlled Iteration Grammars and Full Hyper-AFL's*

PETER R. J. ASVELD

Department of Applied Mathematics, Twente University of Technology, Enschede, The Netherlands

We investigate derivation-controlled K-iteration grammars, called (Γ, K)-iteration grammars, where Γ can be any family of control languages. We prove that already under very weak restrictions on Γ and K the following hold: (i) Regular control does not increase the generating power of K-iteration grammars, (ii) for each (Γ, K)-iteration grammar there exists an equivalent propagating (Γ, K)-iteration grammar, (iii) the family of (Γ, K)-iteration languages is a full hyper-AFL, (iv) for each (Γ, K)-iteration grammar there exists an equivalent (Γ, K)-iteration grammar with exactly two substitutions. We also discuss some additional properties and applications of (uncontrolled) K-iteration grammars and controlled (deterministic) ETOL systems and their languages in a wider context.

INTRODUCTION

Context-independent Lindenmayer systems (L-systems) were originally defined in developmental biology in order to model the parallel nature of growth in arrays of cells (filaments). During the past few years the study of L-systems became a topic in formal language theory (cf. Herman and Rozenberg, 1975).

Rozenberg (1973a, b) introduced tabled (TOL) and extended tabled (ETOL) context-independent L-systems and languages in order to describe the development of filamentous organisms which can develop in a different way under variable physical conditions. Different tables of developmental rules provide for the various physical states of the environment. In the Rozenberg model there are no restrictions on the order of application of the tables, that is, on the sequence of physical states. For instance, if we consider two possible physical states, day (light and warm) and night (dark and cold), then in the TOL and ETOL models a sequence of 3 days followed by 7 nights is allowed.

In this paper we extend the ETOL model in order to be able to describe physical phenomena in a more realistic way. The main idea is that we restrict

* This research has been partly supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.). The main results of this paper have been presented at the Conference on Formal Languages, Automata and Development (organized by A. Lindenmayer and G. Rozenberg) March 31–April 6, 1975, Noordwijkerhout, The Netherlands.

Copyright © 1977 by Academic Press, Inc.
All rights of reproduction in any form reserved.

ISSN 0019–9958

ourselves to certain well-defined sequences of physical states or, equivalently, that we control the order of application of the tables in the ETOL model. This can be done by specifying a language over the "names" of the tables of the ETOL system. The "control language" prescribes the sole order in which we may apply the tables in order to derive a terminal string of the corresponding controlled ETOL language. In this way we can associate with each family of control languages Γ the family (Γ)ETOL, i.e., the family of languages generated by ETOL systems under control languages from Γ.

Returning to our example, in the case that a development starts and ends at day, the control language is the set $\{(\mathbf{d}\,\mathbf{n})^k\mathbf{d} \mid k \geqslant 0\}$ where **d** and **n** are the "names" of the day and night table, respectively. This control language is a regular language and therefore the corresponding controlled ETOL language is a member of the family (REG)ETOL, i.e., the family of regularly controlled ETOL languages.

ETOL systems augmented with control devices have been studied by Nielsen (1975) and, to a certain extent, by Ginsburg and Rozenberg (1975). In both these papers it was shown that regular control does not increase the generating power of ETOL systems and that the family (REG)ETOL thus exactly equals the family ETOL.

Van Leeuwen (1973) and Salomaa (1973b) observed that the finite set of tables of an ETOL system G is nothing but a finite set of finite substitutions over the alphabet of G. Replacing these finite substitutions by arbitrary K-substitutions yields the rather algebraic notion of K-iteration grammar. In this way it is possible to place a part of the theory of L-systems in a more general framework.

In this paper we generalize the definition of K-iteration grammar to the notion of Γ-controlled K-iteration grammar or (Γ, K)-iteration grammar for short. (Γ, K)-iteration grammars are a generalization of (Γ)ETOL systems as K-iteration grammars are of ETOL systems. Results concerning (Γ)ETOL systems are easily obtained from the theory of (Γ, K)-iteration grammars by simply taking K equal to the family of finite languages.

Ginsburg, Greibach, and Hopcroft (1969) introduced the notion of an abstract family of languages (or AFL) in order to investigate in a more abstract sense families of languages closed under certain well-known operations (cf. Ginsburg, 1975). The set of AFL-axioms was extended by Ginsburg and Spanier (1970), leading to the notion of substitution-closed AFL, and Greibach (1970) introduced AFL's closed under nested iterated substitution called super-AFL's.

In the theory of parallel rewriting it turned out to be appropriate to define hyper-AFL's, i.e., AFL's closed under iterated (parallel) substitution (Van Leeuwen, 1973; Salomaa, 1973b). Each hyper-AFL is a super-AFL, whereas each super-AFL in turn is a substitution-closed AFL (Greibach, 1970). However, none of the converse implications holds. Christensen (1974a) proved that ETOL is the smallest full hyper-AFL.

We establish conditions on Γ and K such that the corresponding family of (Γ, K)-iteration languages becomes a full hyper-AFL. More precisely, if K is a prequasoid (i.e., a nontrivial family closed under finite substitution and intersection with regular sets) and if Γ is a nontrivial family closed under union or concatenation, Kleene *, and full marking, then the family of (Γ, K)-iteration languages is a full hyper-AFL containing K, Γ, ETOL, and the family of K-iteration languages. Clearly, the conditions on Γ and K can be very weak; thus, the fact that the family of (Γ, K)-iteration grammars yields a full hyper-AFL depends more on the nature of this grammatical device than on properties of the K-substitutions or of the family of control languages involved.

The paper is organized as follows. Section 1 contains some preliminaries from formal language theory, parallel rewriting, and AFL theory, the definition of (Γ, K)-iteration grammar, and a few examples of (Γ, K)-iteration languages. In Section 2 we prove that regular control does not increase the generating capacity of K-iteration grammars, provided that K contains all languages consisting of exactly one word. In Section 2 we also exhibit an interesting condition on Γ and K such that the family of (Γ, K)-iteration languages equals the family of recursively enumerable languages. In Section 3 we prove a normal form for (Γ, K)-iteration grammars. In Section 4 we prove the main result of the paper and develop a collection of restrictions on Γ and K such that the resulting family of (Γ, K)-iteration languages is a full hyper-AFL. In Section 5 we show that, under weak assumptions, it is possible to construct from any arbitrary (Γ, K)-iteration grammar another (Γ, K)-iteration grammar with only two substitutions generating the same language. Finally, in the last two sections we discuss some applications to (uncontrolled) K-iteration grammars (Section 6), the constructivity of our proofs, and some decision problems for the family of (Γ-controlled) K-iteration languages (Section 7).

1. Definitions

We refer to Salomaa (1973a) or to Hopcroft and Ullman (1969) for all unexplained notation and terminology from formal language theory, and to Herman and Rozenberg (1975) for standard concepts and definitions from the theory of parallel rewriting. AFL theory was introduced by Ginsburg, Greibach, and Hopcroft (1969) and it was recently surveyed by Ginsburg (1975).

Any nonempty collection of languages closed under isomorphism ("renaming of symbols") is called a family. A family K is λ-free when all languages in K are λ-free.

We call a language L nontrivial if L is nonempty and $L \neq \{\lambda\}$. A family K is called nontrivial if K contains a nontrivial language.

In Table I we list the notations and the full names of some language families which we use later. We abbreviate the words "deterministic" and "propagating"

(λ-free) by D and P, respectively. Following this convention, EDTOL denotes the family of deterministic ETOL languages, PDTOL denotes the family generated by propagating deterministic TOL systems, etc.

TABLE I

Notation	Full name	References
SYMBOL	Languages containing exactly one symbol (= a length 1 word)	
ONE	Languages containing exactly one word	
FIN	Finite languages	
REG	Regular languages	Salomaa, 1973a
LCF	Linear context-free languages	Salomaa, 1973a
CF	Context-free languages	Salomaa, 1973a
RE	Recursively enumerable languages	Salomaa, 1973a
SF	Sentential forms of CF	Salomaa, 1973c
OL	O-Lindenmayer languages	Rozenberg and Doucet, 1971*
EOL	Extended OL languages	Herman, 1973, 1974*
TOL	Tabled OL languages	Rozenberg, 1973a*
ETOL	Extended TOL languages	Rozenberg, 1973b*
INDEX	Indexed languages	Aho, 1968; Salomaa, 1973a

* Also see Herman and Rozenberg (1975).

Let K be a family. A K-substitution τ over an alphabet V is a function $\tau\colon V \to K$, such that for all α in V: $\tau(\alpha) \subseteq V^*$. We extend τ in the usual way to words by $\tau(\lambda) = \{\lambda\}$ (λ denotes the empty word), $\tau(\alpha_1 \cdots \alpha_n) = \tau(\alpha_1) \cdots \tau(\alpha_n)$, and to languages by $\tau(L) = \bigcup_{w \in L} \tau(w)$.

If K equals FIN, then we call τ a finite substitution.

The notion of a prequasoid is a slightly weaker variant of the quasoid introduced earlier by Van Leeuwen (1973).

DEFINITION. A family K is a *prequasoid* if

(i) K is nontrivial,

(ii) K is closed under finite substitution,

(iii) K is closed under intersection with regular languages.

A prequasoid K is called a *quasoid* if K contains at least one infinite language.

LEMMA 1.1.

(i) *If K is a prequasoid, then* FIN $\subseteq K$.

(ii) *If K is a quasoid, then* REG $\subseteq K$.

(iii) FIN *is the smallest prequasoid.*

(iv) REG *is the smallest quasoid.*

(v) FIN *is the only prequasoid which is not a quasoid.*

Proof. (i) Let $L \subseteq \Delta^*$ be a nontrivial language in K and let a be a fixed symbol in Δ. Consider the finite substitutions $f: \alpha \mapsto \{\lambda, a\}$, $(\alpha \in \Delta)$ and $g: a \mapsto L_F$, where L_F is an arbitrary finite language. Then clearly $L_F = g(f(L) \cap \{a\})$ is in K.

(ii) Let $L \subseteq \Delta^*$ be an infinite language in K and let $R \subseteq \Sigma^*$ be an arbitrary regular language. Consider the finite substitution $f(a) = \{\lambda\} \cup \Sigma$ for each $a \in \Delta$. Hence $f(L) \cap R = \Sigma^* \cap R = R$ is in K.

The other propositions are now obvious. ∎

A family K is called a semi-AFL when K is closed under union, λ-free homomorphism, inverse homomorphism, and intersection with regular languages. An AFL is a semi-AFL closed under concatenation and Kleene $+$. A (semi-) AFL is a full (semi-)AFL when it is closed under arbitrary homomorphism.

We say that K is closed under left (right) marking when for all languages L in K and all symbols α not occurring in any word of L, the language αL ($L\alpha$) is in K. K is closed under full marking if K is closed under left and right marking. Clearly, marking is a rather weak property: not only are all semi-AFL's closed under marking, but even some well-known anti-AFL's are (i.e., families closed under none of the six AFL operations), e.g., the families OL, TOL, and SF.

In some proofs it turns out to be easier to use a closure under machine mappings that it is to deal with the closure properties originally given.

DEFINITION. A *nondeterministic generalized sequential machine with accepting states* (a-NGSM) is a 6-tuple $T = (Q, \Delta_1, \Delta_2, \delta, q_0, Q_F)$, where (1) Q, Δ_1, and Δ_2 are finite sets (respectively, set of states, input alphabet, and output alphabet); (2) $q_0 \in Q$ is the initial state; (3) $Q_F \subseteq Q$ is the set of final states; (4) δ is a mapping from $Q \times \Delta_1$ into the finite subsets of $Q \times \Delta_2{}^*$. We extend δ to a function from $Q \times \Delta_1{}^*$ into the finite subsets of $Q \times \Delta_2{}^*$ as follows:

(i) $\delta(q, \lambda) = \{(q, \lambda)\}$

(ii) $\delta(q, \omega\alpha) = \{(q', \phi) \mid \phi = \phi_1\phi_2$ and for some q'', $(q'', \phi_1) \in \delta(q, \omega)$ and $(q', \phi_2) \in \delta(q'', \alpha)\}$, where $q \in Q$; $\alpha \in \Delta_1$; $\omega \in \Delta_1{}^*$.

For each a-NGSM $T = (Q, \Delta_1, \Delta_2, \delta, q_0, Q_F)$ the function T from $\Delta_1{}^*$ into the subsets of $\Delta_2{}^*$ defined by $T(\omega) = \{\phi \mid (q, \phi) \in \delta(q_0, \omega)$ for some $q \in Q_F\}$ is called an a-NGSM mapping. We extend the function T in the usual way to languages: $T(L) = \bigcup_{\omega \in L} T(\omega)$.

An a-NGSM $T = (Q, \Delta_1, \Delta_2, \delta, q_0, Q_F)$ is called *deterministic* (an a-GSM) when δ is a mapping from $Q \times \Delta_1$ into $Q \times \Delta_2{}^*$. ∎

It is well-known that a family K is a prequasoid if and only if K is nontrivial and closed under a-NGSM mappings (cf. Hopcroft and Ullman, 1969).

We say that two grammars G_1 and G_2 are equivalent iff their languages are equivalent (modulo the empty word), i.e., they only differ by the empty word: $L(G_1) - \{\lambda\} = L(G_2) - \{\lambda\}$. Two families K_1 and K_2 are equivalent, denoted by $K_1 \equiv K_2$, if for each language L_1 in K_1 there exists an equivalent language L_2 in K_2 and vice versa.

In order to obtain a more general framework to investigate context-independent parallel rewriting, Van Leeuwen (1973) and Salomaa (1973b, 1974a, b) introduced the notion of K-iteration grammar.

DEFINITION. Let K be a family. A *K-iteration grammar* is a 5-tuple $G = (V, \Sigma, U, I, S)$, where V (the alphabet of G) and $\Sigma \subseteq V$ (the terminal alphabet) are finite sets. $S \in V - \Sigma$ is the initial symbol of G. $U = \{\tau_{\mathbf{i}} \mid \mathbf{i} \in I\}$ is a finite set of K-substitutions over V satisfying $\tau_{\mathbf{i}}(\alpha) \neq \varnothing$ for each $\alpha \in V$ and each $\mathbf{i} \in I$. The finite set I is referred to as the index alphabet of G. The language $L(G)$ generated by G is $L(G) = (\bigcup \tau_{\mathbf{i}_n} \cdots \tau_{\mathbf{i}_1}(S)) \cap \Sigma^*$, where the union is taken over all n-tuples $(\mathbf{i}_1, \ldots, \mathbf{i}_n)$ $(n = 0, 1, 2, \ldots)$ with $\mathbf{i}_j \in I$ $(i \leqslant j \leqslant n)$. ∎

We denote the family of languages generated by K-iteration grammars by (K)ITER. $(K)\text{ITER}^{(m)}$ stands for the subfamily of (K)ITER generated by K-iteration grammars where I consists of at most m elements and consequently U contains at most m K-substitutions $(m \geqslant 1)$. By (K)PITER we denote the family of languages generated by K-iteration grammars with only λ-free K-substitutions.

EXAMPLES (Van Leeuwen, 1973; Salomaa, 1973b; Christensen, 1974a):

$$\begin{aligned} (\text{FIN})\text{ITER}^{(1)} &= \text{EOL}; & (\text{FIN})\text{ITER} &= \text{ETOL} \\ (\text{ONE})\text{ITER}^{(1)} &= \text{EDOL}; & (\text{ONE})\text{ITER} &= \text{EDTOL} \\ (\text{ONE})\text{PITER}^{(1)} &= \text{EPDOL}; & (\text{ONE})\text{PITER} &= \text{EPDTOL} \end{aligned}$$

The main idea of a Γ-controlled K-iteration grammar is that we consider the sequence $\mathbf{i}_1 \cdots \mathbf{i}_n$ of indices ("names") of substitutions which are consecutively applied in a derivation as the control word of the derivation. In the definition of Γ-controlled K-iteration languages we only allow sequences of K-substitutions $\mathbf{i}_1 \cdots \mathbf{i}_n$ which belong to a certain given control language $M \subseteq I^*$, with $M \in \Gamma$. Thus, the notion of controlled K-iteration grammar is very similar to corresponding concepts in serial rewritting systems (cf. Ginsburg and Spanier, 1968; Salomaa, 1969, 1970).

Notation. Bold-faced letters are used to indicate terminal symbols and terminal words in control languages.

DEFINITION. Let Γ and K be families of languages. A *(Γ, K)-iteration grammar* is a 6-tuple $G = (V, \Sigma, U, I, M, S)$, where (V, Σ, U, I, S) is a K-

iteration grammar and M is a language such that: (1) $M \subseteq I^*$ and (2) $M \in \Gamma$. We call M the control language of G. The language generated by G is $L(G) = (\bigcup_M \tau_{i_n} \cdots \tau_{i_1}(S)) \cap \Sigma^*$, where the union is taken over all n-tuples $(i_1, \ldots, i_n)$ $(n = 0, 1, 2, \ldots)$ satisfying: $i_1 \cdots i_n \in M$. ∎

(Γ, K)ITER denotes the family of languages generated by (Γ, K)-iteration grammars. (Γ, K)PITER and (Γ, K)ITER$^{(m)}$ $(m \geqslant 1)$ are defined in a completely similar fashion as for uncontrolled K-iteration grammars.

In the terminology of Van Leeuwen (1973) the family (K)ITER is called the hyper-algebraic extension of the family K. Similarly, we may call the family (Γ, K)ITER the Γ-controlled hyper-algebraic extension of the family K. Following Van Leeuwen (1973), a family K is called hyper-algebraically closed iff (K)ITER $= K$, i.e., if K is closed under iterated (parallel) substitution. Obviously, we can consider each K-iteration grammar as a (Γ, K)-iteration grammar with a trivial control language, namely, $M = I^*$.

EXAMPLES. (1) The families ETOL and INDEX are hyper-algebraically closed (cf. Christensen, 1974b; Van Leeuwen, 1973).

(2) Consider $G_1 = (V, \Sigma, U, I, M, S)$ where $V = \Sigma = \{a, b, c\}$, $S = a$, $I = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$; $M = \{\mathbf{a}\,\mathbf{b}^m\,\mathbf{c}\,\mathbf{d}^m \mid m \geqslant 0\}$, and U contains the substitutions

$$\begin{aligned}
&\tau_{\mathbf{a}}(a) = \{ac\}; && \tau_{\mathbf{a}}(b) = \{b\}; && \tau_{\mathbf{a}}(c) = \{c\}\\
&\tau_{\mathbf{b}}(a) = \{aba\}; && \tau_{\mathbf{b}}(b) = \{b\}; && \tau_{\mathbf{b}}(c) = \{c\}\\
&\tau_{\mathbf{c}}(a) = \{aba\}; && \tau_{\mathbf{c}}(b) = \{b\}; && \tau_{\mathbf{c}}(c) = \{c\}\\
&\tau_{\mathbf{d}}(a) = \{a\}; && \tau_{\mathbf{d}}(b) = \{ab\}; && \tau_{\mathbf{d}}(c) = \{ac\}.
\end{aligned}$$

Note that M is in the families SF, PDOL, and LCF and G_1 is a propagating (SF, ONE)-, (PDOL, ONE)-, and (LCF, ONE)-iteration grammar. It is easy to show that

$$L(G_1) = \left(\bigcup_{m=0}^{\infty} \tau_{\mathbf{d}}{}^m \tau_{\mathbf{c}} \tau_{\mathbf{b}}{}^m \tau_{\mathbf{a}}(a)\right) \cap \Sigma^* = \{(a^{m+1}b)^{n-1}\, a^{m+1}c \mid m \geqslant 0;\ n = 2^{m+1}\}.$$

The language $L(G_1)$ is not an OI-Macro language and consequently not in the family INDEX (cf. Fischer, 1968, Aho, 1968).

(3) Consider $G_2 = (V, \Sigma, U, I, M, S)$ where $V = \Sigma = \{a, b\}$, $S = a$, $I = \{\mathbf{a}, \mathbf{b}\}$, $M = \{\mathbf{a}^n\mathbf{b} \mid n \geqslant 0\}$ and U contains the substitutions

$$\begin{aligned}
&\tau_{\mathbf{a}}(a) = \{a^2\}; && \tau_{\mathbf{a}}(b) = \{b\}\\
&\tau_{\mathbf{b}}(a) = b^*ab^*; && \tau_{\mathbf{b}}(b) = \{b\}.
\end{aligned}$$

One can easily show that

(i) $L(G_2) = (\bigcup_{n=0}^{\infty} \tau_{\mathbf{b}} \tau_{\mathbf{a}}{}^n(a)) \cap \Sigma^* = \{x \mid x \in \{a, b\}^*$ and the number of a's in x equals 2^n for some $n \geqslant 0\}$.

(ii) G_2 is a propagating (Γ, K)-iteration grammar with $\Gamma \in \{\text{REG, SF, PDOL}\}$ and $K \in \{\text{REG, POL}\}$.

Herman (1974) proved that $L(G_2)$ is not in the family EOL (cf. Herman and Rozenberg, 1975).

2. Bounds of Control

In this section we consider a few instances of families (Γ, K)ITER where the families Γ and K have members in a restricted domain only. We start by proving that, if K contains all ONE languages, then the family of regular-controlled K-iteration languages is exactly equal to the family generated by K-iteration grammars without control. Thus, regular control does not increase the generating power of K-iteration grammars.

THEOREM 2.1. *If* $\text{ONE} \subseteq K$, *then* $(\text{REG}, K)\text{ITER} = (K)\text{ITER}$.

Proof. The inclusion $(K)\text{ITER} \subseteq (\text{REG}, K)\text{ITER}$ is obvious.

Conversely, let $G = (V, \Sigma, U, I, M, S)$ be a (REG, K)-iteration grammar. Let (Q, I, δ, q_0, Q_F) be a complete deterministic finite state acceptor for M, where Q is the set of states, I is the input alphabet, $\delta: Q \times I \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $Q_F \subseteq Q$ is the set of final states. Let $N_\Sigma = \{A_a \mid a \in \Sigma\}$ be an alphabet of new symbols and let $\psi: V \to (V - \Sigma) \cup N_\Sigma$ be an isomorphism defined by $\psi(\lambda) = \lambda$; $\psi(a) = A_a$ iff a in Σ; $\psi(A) = A$ iff $A \in V - \Sigma$. We extend ψ in the usual way to words and to languages.

Consider the K-iteration grammar $H = (V_0, \Sigma, U_0, I_0, S_0)$, where $V_0 = V \cup Q \cup N_\Sigma \cup \{S_0, F\}$ (F is a rejection symbol, i.e., a nonterminal symbol satisfying $\tau(F) = \{F\}$ for each τ) and $U_0 = \{\tau_{\mathbf{i}}' \mid \mathbf{i} \in I\} \cup \{\tau_{\mathbf{o}}\}$, $I_0 = I \cup \{\mathbf{o}\}$ ($\mathbf{o} \notin I$). For each K-substitution $\tau_{\mathbf{i}}$ in U, we define a corresponding K-substitution $\tau_{\mathbf{i}}'$ in U_0 as follows

$$
\begin{aligned}
\tau_{\mathbf{i}}'(S_0) &= \{q_0 S\} \\
\tau_{\mathbf{i}}'(\alpha) &= \psi(\tau_{\mathbf{i}}(\alpha)) && \text{for each } \alpha \text{ in } V - \Sigma, \\
\tau_{\mathbf{i}}'(A_a) &= \psi(\tau_{\mathbf{i}}(a)) && \text{for each } A_a \text{ in } N_\Sigma \ (a \text{ in } \Sigma), \\
\tau_{\mathbf{i}}'(q) &= \{q'\} && \text{iff } \delta(q, \mathbf{i}) = q' \ (q \text{ in } Q), \\
\tau_{\mathbf{i}}'(\alpha) &= \{F\} && \text{for each } \alpha \text{ in } \Sigma \cup \{F\},
\end{aligned}
$$

and the new K-substitution $\tau_{\mathbf{o}}$ is defined by

$$
\begin{aligned}
\tau_{\mathbf{o}}(q) &= \{\lambda\} && \text{iff } q \text{ in } Q_F, \\
\tau_{\mathbf{o}}(q) &= \{F\} && \text{iff } q \text{ in } Q - Q_F, \\
\tau_{\mathbf{o}}(A_a) &= \{a\} && \text{for each } A_a \text{ in } N_\Sigma \ (a \text{ in } \Sigma), \\
\tau_{\mathbf{o}}(\alpha) &= \{F\} && \text{for each } \alpha \text{ in } V \cup \{S_0, F\}.
\end{aligned}
$$

By the construction of H we obtain: $L(H) = L(G)$, and hence (REG, K)ITER $\subseteq$ (K)ITER. ∎

Theorem 2.1 shows a kind of "lower bound" concerning the effect of control on K-iteration grammars, whereas the following result bears upon the "upper bound" of effective rewriting.

THEOREM 2.2. *If K and Γ are families such that*

(i) ONE $\subseteq K \subseteq$ RE, *and*

(ii) $\{h(L) \mid L \in \Gamma;\ h$ *is an arbitrary homomorphism*$\} =$ RE,

then (Γ, K)ITER = RE.

Proof. Condition (ii) implies that $\Gamma \subseteq$ RE. By Church's thesis we obtain (Γ, K)ITER $\subseteq$ RE, because (Γ, K)-iteration grammars are constructive.

Conversely, let $L \subseteq \Sigma^*$ be an arbitrary recursively enumerable language. Then for $c \notin \Sigma$ the language Lc is also in RE. According to condition (ii) there exists a language $M \subseteq I^*$ in Γ and homomorphism h: $I^* \to (\Sigma \cup \{c\})^*$, such that $h(M) = Lc$.

Consider the (Γ, K)-iteration grammar $G = (V, \Sigma, U, I, M, S)$, where S is a nonterminal symbol; $V = \Sigma \cup \{S\}$, and where for each $\mathbf{i} \in I$ we define $\tau_{\mathbf{i}} \in U$ by

$$\tau_{\mathbf{i}}(S) = \begin{cases} \{h(\mathbf{i})S\} & \text{iff } h(\mathbf{i}) \in \Sigma^*, \\ \{x\} & \text{iff } h(\mathbf{i}) = xc,\ x \text{ in } \Sigma^*, \end{cases}$$

$$\tau_{\mathbf{i}}(\alpha) = \{\alpha\} \qquad \text{iff } \alpha \text{ in } \Sigma.$$

Hence $L(G) = L$, and we obtain RE $\subseteq$ (Γ, K)ITER. ∎

Thus, if K satisfies condition (i) of Theorem 2.2, then the families (Γ, K)ITER, where Γ equals the recursive, the context-sensitive, or even the λ-free context-free programmed languages (Rosenkrantz, 1969), are all equal to the recursively enumerable languages. We could even take Γ to be the intersections of linear languages (cf. Baker and Book, 1974) and still have the same result.

3. NORMAL FORM THEOREM

The main object of this section is to establish a normal form theorem for (Γ, K)-iteration grammars. The definition of normal form is only slightly different from the one for ETOL systems (cf. Rozenberg, 1973b; Herman and Rozenberg, 1975), whereas the proof of this theorem is a simple modification of a result obtained by Rozenberg and Wood (1976), first proved in a simpler form by Van Leeuwen (1973).

DEFINITION. A (Γ-controlled) K-iteration grammar $G = (V, \Sigma, U, I, S)$ ($G = (V, \Sigma, U, I, M, S)$) is in *normal form* if there exist a rejection symbol $F \in V - \Sigma$ and a substitution τ_t in U (called the terminal substitution) which satisfy the following conditions:

(i) for each τ in U and for each α in $\Sigma \cup \{F\}$: $\tau(\alpha) = \{\alpha\}$

(ii) if $\tau \in U - \{\tau_t\}$ and $\alpha \in V - \Sigma - \{F\}$, then $\tau(\alpha) \subseteq (V - \Sigma - \{F\})^+ \cup \{F\}$

(iii) for each α in $V - \Sigma - \{F\}$: $\tau_t(\alpha) = \{\beta\}$ with $\beta \in \Sigma \cup \{F\}$.

The main property of (Γ-controlled) K-iteration grammars in normal form is the fact that all K-substitutions involved are λ-free and that, therefore, these grammars are propagating. In L-systems theory normal forms play a principal part in parsing, complexity considerations (cf. Van Leeuwen, 1975a, 1975b) and rate-of-growth arguments (Asveld and Van Leeuwen, 1975).

The idea in obtaining λ-free K-substitutions to act equivalent to a set of arbitrary K-substitutions is that in each derivation we have to eliminate from the very beginning all occurrences of symbols finally yielding the empty word λ. At each intermediate step in a derivation this is achieved by guessing nondeterministically which occurrences of symbols are unproductive in this sense, and by removing these occurrences from the string. We only keep the set W of symbols that we removed by attaching it to a symbol that we did not remove. This leads to the introduction of a new alphabet $V_1 = \{[\alpha, W] \mid \alpha \in V;\ W \subseteq V\}$.

The new (Γ, K)-iteration grammar H (which should be in normal form and equivalent to a given $G = (V, \Sigma, U, I, M, S)$) must not only simulate the effect of a substitution τ from G on a symbol α occurring in an intermediate stage but also simulate the effect of τ on the symbols stored in the subalphabet W that was attached to α. This leads to a modified substitution τ' over $V_0 = V_1 \cup \Sigma \cup \{F\}$ (F is a rejection symbol) for each substitution τ of G (cf. Rozenberg and Wood, 1976).

THEOREM 3.1 (Normal Form Theorem). *Let K be a prequasoid. If Γ is closed under right marking, then for each (Γ, K)-iteration grammar there exists an equivalent (Γ, K)-iteration grammar in normal form, and consequently* (Γ, K)ITER $\equiv$ (Γ, K)PITER.

Proof (outline). Consider the (Γ, K)-iteration grammar $H = (V_0, \Sigma, U_0, I_0, M_0, S_0)$ where $U_0 = \{\tau_t\} \cup \{\tau' \mid \tau \in U\}$, $I_0 = \{\mathbf{t}\} \cup I$ and $S_0 = [S, \varnothing]$. The K-substitutions τ' are defined in a way similar to the uncontrolled case (cf. Rozenberg and Wood, 1976). Define the new terminal K-substitution τ_t by

$$\begin{aligned} \tau_t([\alpha, \varnothing]) &= \{\alpha\} && \text{iff } \alpha \in \Sigma, \\ \tau_t([\alpha, W]) &= \{F\} && \text{iff } \alpha \notin \Sigma \text{ or } W \neq \varnothing, \\ \tau_t(\beta) &= \{\beta\} && \text{for each } \beta \text{ in } \Sigma \cup \{F\}. \end{aligned}$$

Finally, let $M_0 = M\mathbf{t}$. Then H is in normal form. ∎

Obviously, Theorem 3.1 applies to (Γ)ETOL systems (i.e., (Γ, FIN)-iteration grammars), but not to (Γ)EDTOL systems (i.e., (Γ, ONE)-iteration grammars) because the family ONE is not a prequasoid. However, since a ONE-substitution is nothing but a homomorphism, we can replace in a (Γ)EDTOL system G each τ by a finite number of corresponding ONE-substitutions (one new ONE-substitution τ_W' for each proper subalphabet W of V) in order to obtain an equivalent (Γ)EDTOL system H in normal form. Thus, we have to require that Γ be closed under finite substitution.

THEOREM 3.2. *If Γ is closed under right marking and finite substitution, then for each* (Γ)EDTOL *system there exists an equivalent* (Γ)EDTOL *system in normal form, and consequently* (Γ)EDTOL $\equiv$ (Γ)EPDTOL.

Note that if $G = (V, \Sigma, U, I, M, S)$ is a (Γ, K)-iteration grammar in normal form, then all words in the control language M need not be of the form $\mathbf{wt}$ for some $\mathbf{w}$ in $(I - \{\mathbf{t}\})^*$ or equivalently, $\tau_\mathbf{t}$ need not be the last substitution in every derivation. After the execution of $\tau_\mathbf{t}$, we may apply any number of substitutions from U, because they all leave a terminal word unchanged.

4. FULL HYPER-AFL'S

Van Leeuwen (1973) originally defined a family K to be a (full) hyper-AFL iff K is a (full) AFL which is hyper-algebraically closed, i.e., (K)ITER $= K$, and he showed that each hyper-algebraically closed family which is a quasoid is also a full AFL. But each nontrivial hyper-algebraically closed family closed under finite substitution contains an infinite language and so we can define a full hyper-AFL by

DEFINITION. A family K is a *full hyper-AFL* if

(i) K is a prequasoid,

(ii) K is hyper-algebraically closed, i.e., (K)ITER $= K$.

In this section we prove the main result of this paper, exhibiting a collection of conditions on Γ and K such that the corresponding family of Γ-controlled K-iteration languages is a full hyper-AFL. We always assume that Γ is nontrivial.

We first prove three helpful lemmas, which are of some interest on their own.

LEMMA 4.1. *Let* ONE $\subseteq K$ *and let Γ be closed under right marking. Then* (i) $\Gamma \subseteq (\Gamma, K)$ITER *and* (ii) $K \subseteq (\Gamma, K)$ITER.

Proof. (i) Let $L \subseteq I_0{}^*$ be a language in Γ and let ψ: $I_0{}^* \to \Sigma^*$ be an iso-

morphism ("renaming"). Consider the (Γ, K)-iteration grammar $G = (V, \Sigma, U, I, M, S)$ with $V = \Sigma \cup \{S\}$, $I = I_0 \cup \{\mathbf{c}\}$ $(\mathbf{c} \notin I_0)$, $M = L\mathbf{c}$, and U contains the substitutions

$$\tau_{\mathbf{i}}(S) = \{\psi(\mathbf{i})S\}; \qquad \tau_{\mathbf{i}}(\alpha) = \{\alpha\} \text{ if } \alpha \text{ in } \Sigma, \text{ for each } \mathbf{i} \text{ in } I_0,$$

$$\tau_{\mathbf{c}}(S) = \{\lambda\}; \qquad \tau_{\mathbf{c}}(\alpha) = \{\alpha\} \text{ if } \alpha \text{ in } \Sigma.$$

Now $L(G) = \psi(L)$, which shows that $\Gamma \subseteq (\Gamma, K)$ITER.

To prove (ii), let $L \subseteq \Sigma^*$ be a language in K and let $M_0 \subseteq I_0{}^*$ be an arbitrary nonempty language in Γ. We consider the (Γ, K)-iteration grammar $G = (V, \Sigma, U, I, M, S)$, where $V = \Sigma \cup \{S\}$, $M = M_0\mathbf{c}$, $I = I_0 \cup \{\mathbf{c}\}$ $(\mathbf{c} \notin I_0)$, and U contains the substitutions $\tau_{\mathbf{i}}(\alpha) = \{\alpha\}$ for each $\alpha \in V$ and for each $\mathbf{i} \in I_0$. $\tau_{\mathbf{c}}(S) = L$; $\tau_{\mathbf{c}}(\alpha) = \{\alpha\}$ for each $\alpha \in \Sigma$.

Then $L(G) = L$, and $K \subseteq (\Gamma, K)$ITER. ∎

(K)ITER is obviously included in (Γ, K)ITER if $I^* \in \Gamma$ for all I. We show that this inclusion can also be obtained from the following conditions (cf. Theorem 4.4).

LEMMA 4.2. *Let* SYMBOL $\subseteq K$, *and let* Γ *be closed under* (i) *left or right marking*, (ii) *union or concatenation, and* (iii) *Kleene* *, *Then* (K)ITER $\subseteq$ (Γ, K)ITER.

Proof. We only give a proof for the case that Γ is closed under right marking; the other case is similar.

Let $G = (V, \Sigma, U, I, S)$ be an arbitrary K-iteration grammar with $I = \{\mathbf{1}, \ldots, \mathbf{m}\}$ and let $L \subseteq I_0{}^*$ be a nonempty language in Γ such that $I \cap I_0 = \varnothing$. If Γ is closed under union (concatenation), then the language $M = (L\mathbf{1} \cup L\mathbf{2} \cup \cdots \cup L\mathbf{m})^*$ (or $M = ((L\mathbf{1})^*(L\mathbf{2})^* \cdots (L\mathbf{m})^*)^*$) is in Γ.

Now, consider the (Γ, K)-iteration grammar $H = (V, \Sigma, U_1, I_1, M, S)$, where $U_1 = U \cup \{\tau_{\mathbf{i}} \mid \mathbf{i} \in I_0\}$; $I_1 = I \cup I_0$. For each $\mathbf{i}$ in I_0 the K-substitutions $\tau_{\mathbf{i}}$ are defined by $\tau_{\mathbf{i}}(\alpha) = \{\alpha\}$ for each α in V. This construction yields: $L(G) = L(H)$ is in (Γ, K)ITER. ∎

LEMMA 4.3. *Let* K *be a prequasoid and let* Γ *be closed under full marking. Then* (Γ, K)ITER *is also a prequasoid.*

Proof. Since (Γ, K)ITER is nontrivial, it suffices to prove that (Γ, K)ITER is closed under a-NGSM mappings. Let $G = (V, \Sigma, U, I, M, S)$ be a (Γ, K)-iteration grammar and let $T = (Q, \Sigma, \Delta, \delta, q_0, Q_F)$ be an a-NGSM. We construct a (Γ, K)-iteration grammar H such that $T(L(G)) = L(H)$ and $L(H) \in (\Gamma, K)$ITER.

First, we define a new alphabet $V_1 = \Delta \cup \{S_1, F\} \cup \{[q, \alpha, \bar{q}] \mid q, \bar{q} \in Q; \alpha \in V\}$, where F is a rejection symbol and S_1 is the new initial symbol.

We introduce two new K-substitutions $\tau_{\mathbf{a}}$ and $\tau_{\mathbf{b}}$ over V_1

$$\begin{aligned}
\tau_{\mathbf{a}}(S_1) &= \{[q_0, S, q_f] \mid q_f \in Q_F\} \\
\tau_{\mathbf{a}}(\alpha) &= \{\alpha\} \qquad \text{for each } \alpha \text{ in } V_1 - \{S_1\}, \\
\tau_{\mathbf{b}}(\alpha) &= \{\alpha\} \qquad \text{for each } \alpha \text{ in } \Delta \cup \{S_1, F\}, \\
\tau_{\mathbf{b}}([q, \alpha, \bar{q}]) &= \{w \in \Delta^* \mid (\bar{q}, w) \in \delta(q, \alpha)\} \cup \{F\}.
\end{aligned}$$

For each K-substitution τ in U_0, we define a new corresponding substitution $\bar{\tau}$ over V_1 as follows

$$\bar{\tau}(\alpha) = \{\alpha\} \qquad \text{iff} \quad \alpha \text{ in } \Delta \cup \{S_1, F\}$$

$$\bar{\tau}([q, \alpha, \bar{q}]) = \{[q, \alpha_1, q_1'][q_1', \alpha_2, q_2'] \cdots [q_{n-1}', \alpha_n, \bar{q}] \mid q_1', \ldots, q_{n-1}' \in Q;$$
$$\alpha_1\alpha_2 \cdots \alpha_n \in \tau(\alpha),\ n \geqslant 1\} \cup E(\tau, \alpha, q, \bar{q})$$

for each pair $(q, \bar{q})$ in $Q \times Q$ and each α in V, where $E(\tau, \alpha, q, \bar{q})$ is defined by

$$E(\tau, \alpha, q, \bar{q}) = \begin{cases} \{\lambda\} & \text{iff } \lambda \in \tau(\alpha) \text{ and } q = \bar{q}, \\ \{F\} & \text{iff } \tau(\alpha) = \{\lambda\} \text{ and } q \neq \bar{q}, \\ \varnothing & \text{otherwise.} \end{cases}$$

Distinguish the following two cases:

(i) $\tau(\alpha) = \{\lambda\}$. Then $\bar{\tau}([q, \alpha, \bar{q}]) = E(\tau, \alpha, q, \bar{q})$ is in K.

(ii) $\tau(\alpha) - \{\lambda\} \neq \varnothing$.

Consider for each pair $(q, \bar{q})$ the a-NGSM $T_{q\bar{q}} = (R, \Delta_1, \Delta_2, \delta_1, r_0, R_F)$, where (i) $R = Q$; (ii) $\Delta_1 = V$; (iii) $\Delta_2 = \{[q_1, \alpha, q_2] \mid q_1, q_2 \in Q;\ \alpha \in V\}$; (iv) $r_0 = q$; (v) $R_F = \{\bar{q}\}$; (vi) δ_1 is defined by $\delta_1(q', \alpha_i) = \{(q'', [q', \alpha_i, q''])\} \mid q'' \in Q\}$ for each α_i in V and for each q' in $R = Q$. Thus $T_{q\bar{q}}(\tau(\alpha)) = \bar{\tau}([q, \alpha, \bar{q}])$. Note that $\lambda \in T_{q\bar{q}}(\tau(\alpha))$ iff $\lambda \in \tau(\alpha)$ and $q = \bar{q}$ (i.e. $r_0 \in R_F$).

Since K is a prequasoid we may conclude from (i) and (ii) that $\bar{\tau}$ is a K-substitution over V_1.

Finally, we define the (Γ, K)-iteration grammar $H = (V_1, \Delta, U_1, I_1, M_1, S_1)$ where $I_1 = I \cup \{\mathbf{a}, \mathbf{b}\}$; $M_1 = \mathbf{a}M\mathbf{b}$ and $U_1 = \{\tau_{\mathbf{a}}, \tau_{\mathbf{b}}\} \cup \{\bar{\tau} \mid \tau \in U\}$.

It is easy to prove that $L(H) = T(L(G))$, which implies that (Γ, K)ITER is a prequasoid. ∎

We can now prove the main result.

THEOREM 4.4. *Let K be a prequasoid and let Γ be closed under full marking, union or concatenation, and Kleene *. Then* (Γ, K)ITER *is a full hyper-*AFL *containing* K, Γ, (K)ITER, *and* ETOL.

Proof. The second part of the proposition is a simple consequence of

Lemmas 4.1 and 4.2 and the fact that (FIN)ITER = ETOL. By Theorem 4.3, (Γ, K)ITER is a prequasoid; thus, it only remains to prove that (Γ, K)ITER is hyper-algebraically closed. The inclusion $(\Gamma, K)\text{ITER} \subseteq ((\Gamma, K)\text{ITER})\text{ITER}$ is obvious from the definition of K-iteration grammar.

Conversely, let $G = (V, \Sigma, U, I, S)$ be a (Γ, K)ITER-iteration grammar, where each $\bar{\tau}_\mathbf{i}$ in U is a (Γ, K)ITER-substitution and let $V = \{\alpha_1, \ldots, \alpha_n\}$. For each j $(1 \leqslant j \leqslant n)$ and each $\mathbf{i}$ in I, we assume that $\bar{\tau}_\mathbf{i}(\alpha_j) = L(G_{ij})$ where the $G_{ij} = (V_{ij}, V, U_{ij}, I_{ij}, M_{ij}, S_{ij})$ are (Γ, K)-iteration grammars which have mutually disjoint nonterminal alphabets $V_{ij} - V$.

We may also assume that the G_{ij} satisfy the following conditions:

(i) For each a in V and each τ in U_{ij} : $\tau(a) = \{a\}$.

(ii) If an intermediate string ω (in a derivation according to G_{ij}) contains a symbol from V, then for each τ in U_{ij} : $\tau(\omega) = \{\omega\}$. (Otherwise, we introduce for each a in V a new nonterminal symbol A_a and we replace each occurrence of a in G_{ij} by A_a. Each substitution is extended with $\tau(\alpha) = \{\alpha\}$, $\alpha \in V \cup \{F_0\}$, where F_0 is a new rejection symbol. We introduce a new substitution $\tau_\mathbf{f}$ defined by

$$\begin{aligned} \tau_\mathbf{f}(A_a) &= \{a\} && \text{for each } a \text{ in } V, \\ \tau_\mathbf{f}(a) &= \{a\} && \text{for each } a \text{ in } V \cup \{F_0\}, \\ \tau_\mathbf{f}(\alpha) &= \{F_0\} && \text{for each } \alpha \text{ in } V_{ij} - V. \end{aligned}$$

Finally, we change the control language M_{ij} into $M_{ij}\mathbf{f}$.)

We have to show that $L(G)$ is in (Γ, K)ITER.

We define the (Γ, K)-iteration grammar $H = (V_0, \Sigma, U_0, I_0, M_0, S)$, where $V_0 = \bigcup_{i,j} (V_{ij} \cup \{\bar{S}_{ij}\}) \cup \{F\}$ (F is a rejection symbol and all $\bar{S}_{ij}$ are new nonterminal symbols corresponding to the S_{ij}. Note that $S \in V \subseteq V_{ij} \subseteq V_0$), $U_0 = \{\tau_\mathbf{a}\} \cup \{\tau_\mathbf{i} \mid \mathbf{i} \in I\} \cup \{\tau'_{ijk} \mid \tau_{ijk} \in U_{ij};\ \mathbf{i} \in I;\ 1 \leqslant j \leqslant n\}$ and $I_0 = \{\mathbf{a}\} \cup I \cup \bigcup_{i,j} I_{ij}$. The K-substitutions in U_0 are defined as follows:

(1)
$$\begin{aligned} \tau_\mathbf{a}(\alpha_j) &= \{\bar{S}_{ij} \mid \mathbf{i} \in I\} && \text{iff } \alpha_j \in V, \\ \tau_\mathbf{a}(\alpha) &= \{F\} && \text{iff } \alpha \notin V. \end{aligned}$$

(2) For each $\mathbf{i}$ in I (i.e. the index alphabet of G) $\tau_\mathbf{i}$ is defined by

$$\begin{aligned} \tau_\mathbf{i}(\bar{S}_{ij}) &= \{\bar{S}_{ij}, S_{ij}\} && 1 \leqslant j \leqslant n, \\ \tau_\mathbf{i}(\alpha_j) &= \{\alpha_j\} && \text{iff } \alpha_j \in V, \\ \tau_\mathbf{i}(\alpha) &= \{F\} && \text{iff } \alpha \notin V \cup \{\bar{S}_{ij}\}. \end{aligned}$$

(3) For each substitution τ_{ijk} in U_{ij}, we define

$$\begin{aligned} \tau'_{ijk}(\alpha) &= \tau_{ijk}(\alpha) && \text{iff } \alpha \in V_{ij}, \\ \tau'_{ijk}(\bar{S}_{ip}) &= \{\bar{S}_{ip}\} && 1 \leqslant p \leqslant n, \\ \tau'_{ijk}(\alpha) &= \{F\} && \text{otherwise.} \end{aligned}$$

If the family Γ is closed under union (concatenation) and if $I = \{\mathbf{1},..., \mathbf{m}\}$, then the control language M_0 of H equals $M_0 = (M_\mathbf{1} \cup \cdots \cup M_\mathbf{m})^*$, where for each $\mathbf{i}$ in I $M_\mathbf{i}$ is defined by $M_\mathbf{i} = \mathbf{a}(\mathbf{i}M_{i1} \cup \cdots \cup \mathbf{i}M_{in})^*$ $(M_0 = (M_\mathbf{1}^* \cdots M_\mathbf{m}^*)^*$, where for each $\mathbf{i}$ in I $M_\mathbf{i}$ is defined by $M_\mathbf{i} = \mathbf{a}(\mathbf{i}M_{i1} \cdots \mathbf{i}M_{in})^*)$.

The simulation of G by H proceeds in the following way. Intermediate words generated by H, which correspond to intermediate stages in a derivation according to G, are written over the alphabet V.

Consider the actual simulation of a single $\bar{\tau}_\mathbf{i}$ step ($\bar{\tau}_\mathbf{i}$ in U), which is controlled by the language $M_\mathbf{i}$. A single application of $\tau_\mathbf{a}$ converts all symbols α_j from V into barred initial symbols $\bar{S}_{ij}$. Then $\tau_\mathbf{i}$ checks whether all first indices of these barred initial symbols are indeed equal to i, otherwise at least one rejection symbol is introduced. At the same time some of the occurrences of the barred symbols $\bar{S}_{ij}$ may be changed into their unbarred counterparts S_{ij}, whereas symbols from $\bigcup_{p,j} V_{pj} - V$ change into F.

The symbols S_{ij} initiate an actual derivation according to G_{ij}, i.e. according to the τ'_{ijk} under control M_{ij}. Observe that by the definition of $M_\mathbf{i}$ and $\tau_\mathbf{i}$ different occurrences of S_{ij} may be the objects of different control words from M_{ij}.

Moreover, it is obvious that after the simulation of the $\bar{\tau}_\mathbf{i}$-step only symbols from V will survive a subsequent $\bar{\tau}_{\mathbf{i}'}$-step, i.e., produce a (possibly) terminal substring in the ultimate word.

By a tedious but straightforward argument, one can prove that $L(H) = L(G)$ and hence $((\Gamma, K)\text{ITER})\text{ITER} \subseteq (\Gamma, K)\text{ITER}$, i.e., (Γ, K)ITER is hyper-algebraically closed. ∎

COROLLARY 4.5. *If the conditions of Theorem* 4.4 *are satisfied and* Γ *contains the language* $\{\mathbf{a}\,\mathbf{b}^m\,\mathbf{c}\,\mathbf{d}^m \mid m \geqslant 0\}$ *then the full hyper-*AFL (Γ, K)ITER *properly contains the family* ETOL.

Proof. By Example (2) of Section 1, the language $L(G_1)$ is in the family (Γ, K)ITER but not in the family INDEX. Since ETOL $\subseteq$ INDEX (cf. Culik, 1974) we may conclude that $L(G_1)$ is not an ETOL language. ∎

As one may intuitively expect, it is possible to obtain weaker closure properties under weaker conditions on Γ. In Lemma 4.3 we already showed that (Γ, K)ITER is a prequasoid when Γ is closed under full marking and K is a prequasoid. Similarly, one can establish that if Γ is also closed under concatenation and contains all languages I^* (for each index alphabet I) then (Γ, K)ITER is a full semi-AFL.

Clearly 4.3 and 4.4 apply to (Γ)ETOL but not immediately to (Γ)EDTOL since ONE is not a prequasoid. However, using standard techniques and a modification of the proof of Lemma 4.3 one may show:

THEOREM 4.6. *If* Γ *is closed under full marking, concatenation, Kleene* *, *and*

finite substitution, then (Γ)EDTOL *is closed under union, concatenation, Kleene *, a*-GSM *mappings and, consequently, under arbitrary homomorphism and intersection with regular languages.*

Comparing Theorems 4.4 and 4.6 shows that (Γ)ETOL is a full hyper-AFL, whereas (Γ)EDTOL under even more restrictive conditions on the family Γ need not even be a (full)AFL (e.g., (REG)EDTOL = EDTOL is not closed under inverse homomorphism; cf. Ehrenfeucht and Rozenberg, 1974).

But when Γ satisfies the premises of Theorem 2.2 then (Γ)EDTOL equals the family of recursively enumerable languages, and consequently, it is a full hyper-AFL. This leads to the question whether there exist less trivial full hyper-AFL's generated by controlled EDTOL systems properly contained in the recursively enumerable languages. We shall answer this question affirmatively.

Let Γ be a full hyper-AFL satisfying (Γ)ETOL = Γ; then clearly $\Gamma \subseteq$ (Γ)EDTOL $\subseteq$ (Γ)ETOL = Γ and thus (Γ)EDTOL = Γ is a full hyper-AFL. Asveld and Van Leeuwen (1975) established the existence of a full hyper-AFL K_ω, which equals the least full hyper-AFL Γ, such that (Γ)ETOL = Γ. Moreover, it was shown that K_ω is properly contained in the context-sensitive languages. Hence, the family of languages generated by K_ω-controlled EDTOL systems is a full hyper-AFL with (K_ω)EDTOL = K_ω, which is properly contained in the context-sensitive languages.

A strongly related but more general problem is the following: Do there exist trade-offs between (non)determinism and control, i.e., can we find for any given family (Γ)ETOL (which is possibly a full hyper-AFL) another family Γ_0 containing Γ with (Γ)ETOL = (Γ_0)EDTOL?

5. On the Number of Substitutions

In this section we show that under certain conditions we can simulate each (Γ, K)-iteration grammar by a (Γ, K)-iteration grammar with only two substitutions. The construction is a generalization of a similar result for ETOL systems due to Rozenberg (1973b).

First, we introduce a class of λ-free homomorphisms. Let I be an index alphabet containing m symbols ($m \geqslant 2$): $I = \{\mathbf{i}_1, \ldots, \mathbf{i}_m\}$. If I_0 is an index alphabet containing exactly two symbols, $I_0 = \{\mathbf{a}, \mathbf{b}\}$, then we define the homomorphism h: $I^* \to I_0^*$ by $h(\mathbf{i}_k) = \mathbf{a}^k\mathbf{b}$ for each k ($1 \leqslant k \leqslant m$). h is extended in the usual way to words and to languages.

Theorem 5.1. *Let* SYMBOL $\subseteq K$ *and let* Γ *be closed under* h. *Then* (Γ, K)ITER$^{(2)}$ = (Γ, K)ITER$^{(m)}$ = (Γ, K)ITER *for each* $m \geqslant 2$.

Proof. Obviously (Γ, K)ITER$^{(2)} \subseteq$ (Γ, K)ITER$^{(m)} \subseteq$ (Γ, K)ITER ($m \geqslant 2$).

Conversely, let $G = (V, \Sigma, U, I, M, S)$ be a (Γ, K)-iteration grammar where $I = \{\mathbf{i}_1, \ldots, \mathbf{i}_m\}$ for some $m \geqslant 3$ (the cases $m = 1$ and $m = 2$ are trivial).

Define for each j with $1 \leqslant j \leqslant m$ an isomorphism ("renaming of symbols") ψ_j by $\psi_j(\alpha) = \alpha_j$ (α in V; all α_j's are new symbols) and extend these isomorphisms in the usual way to words and to languages. Define a new alphabet by $V_0 = V \cup \{F\} \cup \{\psi_j(\alpha) \mid \alpha \in V;\ 1 \leqslant j \leqslant m\}$ (F is a rejection symbol).

Let $\tau_{\mathbf{a}}$ and $\tau_{\mathbf{b}}$ be K-substitutions, respectively defined by

$$\begin{aligned}
\tau_{\mathbf{a}}(\alpha) &= \{\alpha_1\} && \text{for each } \alpha \text{ in } V,\\
\tau_{\mathbf{a}}(\alpha_j) &= \{\alpha_{j+1}\} && \text{for each } \alpha \text{ in } V \text{ and each } j\ (1 \leqslant j \leqslant m-1),\\
\tau_{\mathbf{a}}(\beta) &= \{F\} && \text{for each } \beta \text{ in } \{F\} \cup \{\psi_m(\alpha) \mid \alpha \in V\},\\
\tau_{\mathbf{b}}(\alpha_j) &= \tau_{\mathbf{i}_j}(\alpha) && \text{for each } \alpha \text{ in } V \text{ and each } j\ (1 \leqslant j \leqslant m),\\
\tau_{\mathbf{b}}(\beta) &= \{F\} && \text{for each } \beta \text{ in } V \cup \{F\}.
\end{aligned}$$

Finally, let $H = (V_0, \Sigma, U_0, I_0, M_0, S)$ be a (Γ, K)-iteration grammar where $I_0 = \{\mathbf{a}, \mathbf{b}\}$, $U_0 = \{\tau_{\mathbf{a}}, \tau_{\mathbf{b}}\}$ and $M_0 = h(M)$.

A simulation of $\tau_{\mathbf{k}}$ of G by H proceeds in the following way: (i) At the start of the simulation of $\tau_{\mathbf{k}}$ the intermediate result is a word over V. (ii) Applying k times the K-substitution $\tau_{\mathbf{a}}$ of H as forced by the new control provides all symbols with a subscript k. (iii) Next, execution of $\tau_{\mathbf{b}}$ performs the actual simulation of $\tau_{\mathbf{k}}$, while all subscripts are removed.

By the construction of H we obtain $L(H) = L(G)$, and hence, $(\Gamma, K)\text{ITER} \subseteq (\Gamma, K)\text{ITER}^{(m)} \subseteq (\Gamma, K)\text{ITER}^{(2)}$ for each $m \geqslant 2$. ∎

Note, that in the construction of H the substitution $\tau_{\mathbf{a}}$ is a symbol-to-symbol substitution rather than an arbitrary K-substitution.

As a direct consequence of Theorem 5.1, it is possible to reduce any number of tables of a (Γ)ETOL or (Γ)EDTOL system to exactly two tables, provided that Γ satisfies the condition of Theorem 5.1.

6. Uncontrolled Iteration Languages

In this section we discuss some simple applications to the families generated by uncontrolled K-iteration grammars.

First, we consider a Normal Form Theorem for K-iteration grammars.

Corollary 6.1. *Let K be a prequasoid. Then for each K-iteration grammar there exists an equivalent K-iteration grammar in normal form, and consequently* $(K)\text{ITER} \equiv (K)\text{PITER}$.

Similar results have been obtained by Van Leeuwen (1973), Salomaa (1973b, 1974a, b) and by Rozenberg and Wood (1976).

Salomaa (1973b) investigated the families $(K)\text{ITER}^{(m)}$ (for $m \geqslant 1$) and (K)ITER; they are all full AFL's, provided that K is a quasoid. We show that for $m \geqslant 2$ all these families are identical (Corollary 6.2) and that they form a full hyper-AFL, even if K is a prequasoid rather than a quasoid (Theorem 6.3).

COROLLARY 6.2. *If* SYMBOL $\subseteq K$, *then for each* $m \geqslant 2$: $(K)\text{ITER}^{(2)} = (K)\text{ITER}^{(m)} = (K)$ITER.

Proof. When we remove the control in the proof of Theorem 5.1 or, equivalently, when we take $M = I^*$ and $M_0 = I_0{}^* = \{\mathbf{a}, \mathbf{b}\}^*$, one easily verifies that $L(H) = L(G)$. This implies

$$(K)\text{ITER} \subseteq (K)\text{ITER}^{(m)} \subseteq (K)\text{ITER}^{(2)} \qquad (m \geqslant 2),$$

whereas the converse inclusions are obvious. ∎

Corollary 6.2 is the "best possible" result, i.e., one can in general not reduce the number of substitutions to one (cf. Rozenberg, 1973b).

The following theorem is the main result of this section.

THEOREM 6.3. *Let* K *be a prequasoid. Then* (K)ITER *is a full hyper-*AFL *containing* ETOL. *Moreover,* (K)ITER *is the smallest full hyper-*AFL *containing* K.

Proof. By Theorems 2.1 and 4.4, (K)ITER is a full hyper-AFL containing ETOL.

Now, let K_0 be a full hyper-AFL containing the family K: $K \subseteq K_0$. Then clearly $(K)\text{ITER} \subseteq (K_0)\text{ITER}$ and since K_0 is hyper-algebraically closed we obtain $(K)\text{ITER} \subseteq K_0$. ∎

Theorem 6.3 is the essential generalization of the fact that ETOL is the smallest full hyper-AFL (Christensen, 1974a, b).

Observe that $(K)\text{ITER}^{(1)}$, where K is a prequasoid rather than a quasoid, need not be a full AFL. As an example we note that $(\text{FIN})\text{ITER}^{(1)} = \text{EOL}$ is not closed under inverse homomorphism (Herman, 1974, or Herman and Rozenberg, 1975).

7. CONSTRUCTIVITY AND DECIDABILITY

Let $L_1, \ldots, L_n$ be languages in an arbitrary family K and let $G_1, \ldots, G_n$ be rewriting systems (grammars) satisfying $L_i = L(G_i)$ for each i $(1 \leqslant i \leqslant n)$. If f is an n-ary operation on languages, then we say that K is effectively closed under f if and only if there exists an algorithm that given $G_1, \ldots, G_n$ will produce a rewriting system (grammar) G, such that $L(G) = f(L_1, \ldots, L_n)$ and $L(G) \in K$.

As usual, we say that a proof which shows that the family K is closed under f is constructive when that proof exhibits such an algorithm.

Except in the case of Theorem 3.1 and Lemma 4.3, all proofs in this paper are constructive (provided that we replace the phrase "closed" in the conditions by "effectively closed" everywhere).

The proof of Theorem 3.1 turns out to be constructive if the emptiness problem is decidable for languages in K and if K is effectively closed under intersection with regular languages (cf. Rozenberg and Wood, 1976; Wood, 1975).

In the proof of Lemma 4.3 the crucial point is the construction of $E(\tau, \alpha, q, \bar{q})$. This set can be obtained in a constructive way, again provided that the emptiness problem is decidable in K and that K is effectively closed under intersection with regular languages.

Thus, if we restrict our attention to effective closure properties, the decidability of the emptiness problem in K implies the constructivity of all proofs in this paper. Since the emptiness problem is trivially decidable when K equals FIN or ONE, all the corresponding proofs in the (Γ)ETOL and (Γ)EDTOL case are constructive.

We conclude this paper with a few results on decidability problems. In the remaining part of this section we only consider effective closure properties.

We need the definition of a Szilard language for a K-iteration grammar (cf. Wood, 1975), which is the collection of all sequences of substitutions which can finally yield terminal words in derivations.

DEFINITION. Let $G = (V, \Sigma, U, I, S)$ be a K-iteration grammar; then the *Szilard language* of G (denoted by $\mathrm{Sz}(G)$) is

$$\mathrm{Sz}(G) = \{\mathbf{i}_1 \cdots \mathbf{i}_n \mid \tau_{\mathbf{i}_n} \cdots \tau_{\mathbf{i}_1}(S) \cap \Sigma^* \neq \varnothing, n \geqslant 1 \quad \text{for some } n\text{-tuple}$$
$$(\mathbf{i}_1, \ldots, \mathbf{i}_n) \text{ with } \mathbf{i}_j \in I,\ 1 \leqslant j \leqslant n\}.$$

The Szilard languages of K-iteration grammars have been studied by Wood (1975):

LEMMA 7.1. *If G is a K-iteration grammar, then the Szilard language* $\mathrm{Sz}(G)$ *is a regular language.*

This lemma is constructive under the same conditions and for the very same reason as Theorem 3.1 is (cf. Wood, 1975; Rozenberg and Wood, 1976).

THEOREM 7.2. *Let Γ and K be closed under intersection with regular languages and let the emptiness problem be decidable for languages in Γ and K. Then the emptiness problem is decidable in* (Γ, K)ITER.

Proof. Let $G = (V, \Sigma, U, I, M, S)$ be a (Γ, K)-iteration grammar and consider the corresponding uncontrolled K-iteration grammar $G_0 = (V, \Sigma, U, I, S)$. By Lemma 7.1, $\mathrm{Sz}(G_0)$ is a regular language and therefore $M \cap \mathrm{Sz}(G_0)$ is in Γ. Moreover, by the definition of $\mathrm{Sz}(G_0)$, $L(G_0) = \varnothing$ if and only if $\mathrm{Sz}(G_0) = \varnothing$.

Hence $L(G) = \varnothing$ if and only if $M \cap \mathrm{Sz}(G_0) = \varnothing$. ∎

COROLLARY 7.3. *Let K be a prequasoid and let Γ be closed under full marking and intersection with regular languages. If the emptiness problem is decidable in Γ and K, then the membership problem is decidable for* (Γ, K)ITER.

Proof. By Lemma 4.3, the family (Γ, K)ITER is closed under intersection with regular languages. So we can construct a (Γ, K)-iteration grammar H with $L(H) = L(G) \cap \{w\}$ for each word w and each (Γ, K)-iteration grammar G. Hence, if suffices to consider the emptiness problem with respect to H. ∎

COROLLARY 7.4. *If K is closed under intersection with regular languages and if the emptiness problem is decidable in K, then the emptiness problem is also decidable in* (K)ITER.

COROLLARY 7.5. *Let K be a prequasoid and let the emptiness problem be decidable in K. Then the membership problem in* (K)ITER *is decidable.*

By the trivial decidability of the emptiness problem in the family FIN, we directly obtain:

COROLLARY 7.6. *Let Γ be closed under intersection with regular languages and let the emptiness problem be decidable in Γ. Then the emptiness problem is decidable in* (Γ)ETOL.

COROLLARY 7.7. *Let Γ be closed under intersection with regular languages and let the emptiness problem be decidable for languages in Γ. Then the membership problem is decidable in* (Γ)ETOL.

Proof. Let $G = (V, \Sigma, U, I, M, S)$ be a (Γ)ETOL system. Ginsburg and Rozenberg (1975) proved that the language

$$\{\mathbf{i}_1 \cdots \mathbf{i}_n \in I^* \mid \tau_{\mathbf{i}_n} \cdots \tau_{\mathbf{i}_1}(S) \cap R \neq \varnothing;\ \tau_{\mathbf{i}_j} \in U \text{ for each } j\ (1 \leqslant j \leqslant n)\}$$

is regular for each regular set R.

Since Γ is closed under intersection with regular languages, taking R equal to $\{w\}$ (for some terminal word w), reduces the membership problem for (Γ)ETOL to the emptiness problem for Γ (cf. proof of Corollary 7.3). ∎

ACKNOWLEDGMENT

I am grateful to Joost Engelfriet, Jan van Leeuwen, and Leo Verbeek for many fruitful discussions and suggestions concerning the subject and for their critical reading of an earlier draft of this paper. Thanks are also due to the referee for indicating numerous changes to improve the presentation.

RECEIVED: January 30, 1976; REVISED: September 13, 1976

REFERENCES

AHO, A. V. (1968), Indexed grammars—An extension of context-free grammars, *J. Assoc. Comp. Mach.* **15**, 647–671.

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.

CHRISTENSEN, P. A. (1974a), Hyper-AFL's and ETOL Systems, DAIMI PB-35, University of Aarhus, Aarhus, Denmark.

CHRISTENSEN, P. A. (1974b), Hyper-AFL's and ETOL Systems, *in* "*L* Systems" (G. Rozenberg and A. Salomaa, Eds.), pp. 254–257, Springer–Verlag, Berlin.

CULIK, K., II (1974), On some families of languages related to developmental systems, *Internat. J. Comput. Math.* **4**, 31–42.

EHRENFEUCHT, A., AND ROZENBERG, G. (1974), Three useful results concerning *L* languages without interactions, *in* "*L* Systems" (G. Rozenberg and A. Salomaa, Eds.), pp. 72–77, Springer–Verlag, Berlin.

FISCHER, M. J. (1968), Grammars with macro-like productions, *in* "IEEE Conference Record of 1968, 9th Annual Symposium on Switching and Automata Theory," 131–142.

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

GINSBURG, S., GREIBACH, S. A., AND HOPCROFT, J. E. (1969), Studies in abstract families of languages, *Mem. Amer. Math. Soc.* **87**.

GINSBURG, S., AND ROZENBERG, G. (1975), TOL schemes and control sets, *Inform. Contr.* **27**, 109–125.

GINSBURG, S., AND SPANIER, E. H. (1968), Control sets on grammars, *Math. Systems Theory* **2**, 159–177.

GINSBURG, S., AND SPANIER, E. H. (1970), Substitution in families of languages, *Inform. Sci.* **2**, 83–110.

GREIBACH, S. A. (1970), Full AFL's and nested iterated substitution, *Inform. Contr.* **16**, 7–35.

HERMAN, G. T. (1973), A biologically motivated extension of ALGOL-like languages, *Inform. Contr.* **22**, 487–502.

HERMAN, G. T. (1974), Closure properties of some families of languages associated with biological systems, *Inform. Contr.* **24**, 101–121.

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.

NIELSEN, M. (1975), EOL systems with control devices, *Acta Informatica* **4**, 373–386.

ROSENKRANTZ, D. J. (1969), Programmed grammars and classes of formal languages, *J. Assoc. Comp. Mach.* **16**, 107–131.

ROZENBERG, G. (1973a), TOL systems and languages, *Inform. Contr.* **23**, 357–381.

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

ROZENBERG, G., AND DOUCET, P. G. (1971), On OL-languages, *Inform. Contr.* **19**, 302–318.

ROZENBERG, G., AND WOOD, D. (1976), A note on K-iteration grammars, *Inform. Processing Letters* **4**, 162–164.

SALOMAA, A. (1969), On grammars with restricted use of productions, *Ann. Acad. Sci. Fennicae Ser. AI* **454**.

SALOMAA, A. (1970), On some families of formal languages obtained by regulated derivations, *Ann. Acad. Sci. Fennicae Ser. AI* **479**.

SALOMAA, A. (1973a), "Formal Languages," Academic Press, New York.

SALOMAA, A. (1973b), Macros, iterated substitution and Lindenmayer AFL's, DAIMI PB-18, University of Aarhus, Aarhus, Denmark.

SALOMAA, A. (1973c), On sentential forms of context-free grammars, *Acta Informatica* **2**, 40–49.

SALOMAA, A. (1974a), Iteration grammars and Lindenmayer AFL's, *in* "L Systems" (G. Rozenberg and A. Salomaa, Eds.), pp. 250–253, Springer–Verlag, Berlin.

SALOMAA, A. (1974b), Parallelism in rewriting systems, *in* "Automata Languages and Programming, 2nd Colloquium" (J. Loeckx, Ed.), pp. 523–533, Springer–Verlag, Berlin.

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

VAN LEEUWEN, J. (1975a), The tape-complexity of context-independent developmental languages, *J. Comp. System Sci.* **11**, 203–211.

VAN LEEUWEN, J. (1975b), Deterministically Recognizing EOL-languages in Time $O(n^{3.81})$, Report IN 9/75, Mathematical Centre, Amsterdam.

WOOD, D. (1975), A note on Lindenmayer systems, Szilard languages, spectra, and equivalence, *Internat. J. Comp. Inform. Sci.* **4**, 53–62.

Printed by the St Catherine Press Ltd., Tempelhof 37, Bruges, Belgium

Chapter 3

Iterated Deterministic Substitution

with *Joost Engelfriet*

Reprinted from *Acta Informatica 8 (1977) 285 - 302*

Remark

In this paper a family is not required to be closed under isomorphism.

The hyper-algebraic extension of a family K is denoted in the usual way, viz. by H(K).

Iteration grammars are defined in a way that empty languages may occur in substitutions.

Erratum

p. 68 (292) line 35: change "η_1(ONE=EDOL" to "η_1(ONE)=EDOL".

Acta Informatica 8, 285 – 302 (1977)

Acta Informatica

© by Springer-Verlag 1977

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 η(CF) is properly included in IO, and that η(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 $\bigcup K = \{x \mid x \in X$ for some X 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. ONE $= \{\{\omega\} \mid \omega \in V_\infty^*\}$. The family SYMBOL is the subfamily of ONE defined by SYMBOL $= \{\{\sigma\} \mid \sigma \in V_\infty\}$. 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 $\tau(\sigma_1 \dots \sigma_n) = \{\omega_1 \dots \omega_n \mid \omega_i \in \tau(\sigma_i)\}$), and extended to languages L over V by $\tau(L) = \bigcup \{\tau(\omega) \mid \omega \in L\}$. 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 $\omega \in V^*$, $\tau(\{\omega\}) = \{h(\omega) \mid h$ is a homomorphism such that $h(\sigma) \in \tau(\sigma)$ for each $\sigma \in V\}$. For arbitrary languages over V, $\tau(L) = \bigcup \{\tau(\{\omega\}) \mid \omega \in L\}$. □

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 $\tau(\{\sigma_1 \dots \sigma_n\}) = \{\omega_1 \dots \omega_n \mid \omega_i \in \tau(\sigma_i)$, 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, $\tau(\{ababa\})=\{a^nba^nba^n|n\geqq 0\}\cup\{a^nbba^nbba^n|n\geqq 0\}$; however with τ viewed as a REG-substitution, $\tau(\{ababa\})=\{a^kxa^mya^n|k,m,n\geqq 0$ 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. □

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

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 dK-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 dK-substitutions over V, indexed by I.

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

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

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 dK-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 $m \geqq 1$, $\eta_m(K)$ denotes the family of all languages generated by dK-iteration grammars that have at most m d-substitutions. One should note that $\eta(K)$ does not always include K; however, if for instance 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 $L(G)=\{a^{2^n 3^m} | n, m \geqq 0\} \cup \{\lambda\}$. 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 dK-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 $\tau_p(S)=\{\#a\#aS, \lambda\}$ 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 $L(G)=\{\#w_1\#w_1\#w_2\#w_2 \ldots \#w_n\#w_n | n \geqq 1$ and $w_i \in \tau_q(a)\}$. □

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 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, $H(\mathrm{FIN})=H(\mathrm{FIN}-\{\emptyset\})$ and $\eta(\mathrm{ONE} \cup \{\emptyset\})=\eta(\mathrm{ONE})$. 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) $H_1(\mathrm{ONE})=\eta_1(\mathrm{ONE})=\mathrm{EDOL}$ and $H_1(\mathrm{FIN})=\mathrm{EOL}$.

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

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

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

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

Proof. By Example 2.3(2) we have to show that $\eta(\mathrm{FIN})=\eta(\mathrm{ONE})$. Since obviously $\eta(\mathrm{ONE}) \subseteq \eta(\mathrm{FIN})$, it suffices to show that $\eta(\mathrm{FIN}) \subseteq \eta(\mathrm{ONE})$. To prove this, let $G=(V, T, I, U, S)$ be a dFIN-iteration grammar with $U=\{\tau_i | i \in I\}$. 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

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

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

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

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: $\mathrm{EDOL}=\eta_1(\mathrm{ONE})$ is properly included in $\eta_1(\mathrm{FIN})$. In fact $\eta_1(\mathrm{FIN})$ 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 $\phi \overset{i}{\Rightarrow} \psi$ if $\psi \in \tau_i(\{\phi\})$. For $\pi \in I^*$, we write $\phi \overset{\pi}{\Rightarrow} \psi$ if $\psi \in \tau_{i_n}(\ldots \tau_{i_2}(\tau_{i_1}(\{\phi\}))\ldots)$, where $\pi=i_1 i_2 \ldots i_n$ (in particular, if $\pi=\lambda$, then

$\phi \overset{\pi}{\Rightarrow} \psi$ iff $\phi = \psi$). Finally we write $\phi \overset{*}{\Rightarrow} \psi$ if $\phi \overset{\pi}{\Rightarrow} \psi$ for some $\pi \in I^*$. If necessary, the symbol G is added as a subscript to $\Rightarrow$. As usual, a sequence $\phi_1, \ldots, \phi_n$ such that for all j $(1 \leqq j \leqq n-1)$ $\phi_j \overset{i}{\Rightarrow} \phi_{j+1}$ for some $i \in I$, is called a derivation from ϕ_1 to ϕ_n. Obviously $\phi \overset{*}{\Rightarrow} \psi$ iff there is a derivation from ϕ to ψ. Note that, as usual, $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$.

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 $\sigma_1, \ldots, \sigma_n \in V$, $\omega \in V^*$ and $\pi \in I^*$; then $\sigma_1 \sigma_2 \ldots \sigma_n \overset{\pi}{\Rightarrow} \omega$ iff there exist $\omega_1, \omega_2, \ldots, \omega_n \in V^*$ such that $\omega = \omega_1 \omega_2 \ldots \omega_n$ and $\sigma_k \overset{\pi}{\Rightarrow} \omega_k$ for all k, $1 \leqq k \leqq n$. For d-iteration grammars a similar property can obviously be stated for one-step derivations (assuming that all $\tau_i(\sigma)$ are nonempty): let $\sigma_1, \ldots, \sigma_n \in V$, $\omega \in V^*$ and $i \in I$; then $\sigma_1 \sigma_2 \ldots \sigma_n \overset{i}{\Rightarrow} \omega$ iff there exist $\omega_1, \ldots, \omega_n \in V^*$ such that (1) $\omega = \omega_1 \ldots \omega_n$, (2) $\sigma_k \overset{i}{\Rightarrow} \omega_k$ for all k, $1 \leqq k \leqq n$, and (3) if $\sigma_k = \sigma_m$ then $\omega_k = \omega_m$ $(1 \leqq k, m \leqq n)$. 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 dΓK-iteration grammar) is a 6-tuple $G = (V, T, I, U, M, S)$ where (V, T, I, U, S) is a dK-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 $\{w \in T^* \mid S \overset{\pi}{\Rightarrow} w$ for some $\pi \in M\}$. □

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 dΓK-iteration grammars. For $m \geqq 1$, $\eta_m(\Gamma, K)$ denotes the family of languages generated by dΓK-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 dK-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 dK-iteration grammar with M_1

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

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 $H(\Gamma, \mathrm{ONE})=\eta(\Gamma, \mathrm{ONE})$; these are the Γ-controlled EDTOL languages of [1]. Finally, if Γ is closed under finite substitution, then $\eta(\Gamma, \mathrm{FIN})=\eta(\Gamma, \mathrm{ONE})$. 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 $[i, \omega_1, \ldots, \omega_n]$ 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* $\mathrm{ONE}\subseteq K$. *Then* $\eta(\mathrm{REG}, K)=\eta(K)$ *and* $H(\mathrm{REG}, K)=H(K)$. □

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) $\{h(M)|M\in\Gamma$ *and h is a homomorphism*$\}=\mathrm{RE}$, *and* (2) $\mathrm{ONE}\subseteq K\subseteq\mathrm{RE}$. *Then* $\eta(\Gamma, K)=H(\Gamma, K)=\mathrm{RE}$. □

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* $H_2(K)=H(K)$. □

Note that in general the number of d-substitutions cannot be reduced to one; for instance, $H_1(\mathrm{ONE})=\eta_1(\mathrm{ONE}=\mathrm{EDOL}$ is properly included in EDTOL $=\eta(\mathrm{ONE})=H(\mathrm{ONE})$, 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 dΓK-iteration grammar there exists a λ-free dΓK-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 $alph(\omega)$ denotes the set of all symbols occurring in ω. Note that $alph(\lambda)=\emptyset$. The function $alph$ is extended to languages L by $alph(L)=\bigcup\{alph(\omega)|\omega\in L\}$.

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 $alph(\omega_1)=alph(\omega_2)$.*

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 $alph(\omega)$) may be removed by replacing each d-substitution by finitely many new ones. Let $G=(V, T, I_0, U_0, S)$ with $U=\{\tau_i|i\in I_0\}$. For each $X\subseteq V$ we define the regular language $R(X)=\{\omega\in V^*|alph(\omega)=X\}$. Note that $R(\emptyset)=\{\lambda\}$. Now, for each language L over V, $L=\bigcup\{L\cap R(X)|X\subseteq V\}$. 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 $\{[i, X_1, \ldots, X_n]|i\in I_0$ and $X_k\subseteq V$ for all k, $1\leqq k\leqq n\}$, where $V=\{\sigma_1, \ldots, \sigma_n\}$. Then we define, for $j=[i, X_1, \ldots, X_n]$, $\tau_j\in U$ 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 $f(i)=\{[i, X_1, \ldots, X_n]|X_k\subseteq V, 1\leqq k\leqq n\}$ for each $i\in I$. It should now be clear that $L(H)=L(G)$, and a formal proof is left to the reader. □

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 dΓK-iteration grammar, and let $H=(V, T, I, U, M, S)$ be a dΓK-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 dΓK-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 $alph(\tau(W))$ have to be in

the set W' of unproductive symbols of the next stage (and note that if $\tau(\sigma)=\{\lambda\}$ then $alph(\tau(\sigma))=\emptyset$ so that unproductiveness of σ is indeed checked). Thus W' will in fact be of the form $X\cup alph(\tau(W))$, 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 $[\sigma_1, W][\sigma_2, W]\ldots[\sigma_n, W]$ with $\{\sigma_1,\ldots,\sigma_n\}\cap W=\emptyset$.

The formal construction is as follows. Let $T_N=T$, and $V_N=T\cup\{F\}\cup\{[\sigma,W]|\ \sigma\in V$ 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 prober 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

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

Let τ_t be the dK-substitution defined by

$$\tau_t([\sigma,\emptyset])=\{\sigma\}\quad \text{for } \sigma\in T,$$
$$\tau_t([\sigma,W])=\{F\}\quad \text{if } \sigma\notin T \text{ or } W\neq\emptyset,$$
$$\tau_t(\sigma)=\{\sigma\}\quad \text{for } \sigma\in T\cup\{F\}.$$

Now we define $I_N=\{(i,X)|i\in I$ and $X\subsetneq V\}\cup\{t\}$. Finally let g be the finite substitution such that, for $i\in I$, $g(i)=\{(i,X)|X\subsetneq V\}$, 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 dΓK-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 $[\sigma_1\ldots\sigma_n, W]$ as an abbreviation of $[\sigma_1, W]\ldots[\sigma_n, W]$.

(1) For $\omega\in V^*$, $z\in T^+$ and $\pi\in I^*$, if $\omega\overset{\pi}{\underset{H}{\Rightarrow}}z$, then $[h_W(\omega), W]\overset{ut}{\underset{G_N}{\Rightarrow}}z$ for some $u\in g(\pi)$, where W is the set of all symbols of ω that generate λ (in the derivation $\omega\overset{\pi}{\underset{H}{\Rightarrow}}z$) 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 $[\omega, W]\overset{ut}{\underset{G_N}{\Rightarrow}}z$, then $\psi\overset{\pi}{\underset{H}{\Rightarrow}}z$ 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 $L(G_N)=L(H)-\{\lambda\}$. □

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 $L(G'_N) = L(G_N) = L(G) - \{\lambda\}$. □

Since EPDOL $\subsetneq$ 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. $\eta(K) \subseteq K$. □

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 dΓK-iteration grammar and $h: T^* \to T_0^*$ a homomorphism. We may obviously assume that $V \cap T_0 = \emptyset$. Construct the dΓK-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 dΓK-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 $\delta: Q \times T \to Q$ 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 $\phi: Q \to Q$ 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 d$\Gamma K$-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 $R_\phi=\{[\sigma_1, \phi_1][\sigma_2, \phi_2] \ldots [\sigma_n, \phi_n] \mid n \geqq 1$ and $\phi_n \circ \cdots \circ \phi_2 \circ \phi_1 = \phi\} \cup E_\phi$, 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 $f: V \to \Phi$ the homomorphism $f': V^* \to (V \times \Phi)^*$ by $f'(\sigma)=[\sigma, f(\sigma)]$ for each $\sigma \in V$.

Now let $I_1=\{(i, f) \mid i \in I$ and $f: V \to \Phi\} \cup \{\phi \in \Phi \mid \phi(q_0) \in Q_F\} \cup \{t\}$. For each $i \in I$ and $f: V \to \Phi$ 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

$g(s)=\{\phi\in\Phi\mid\phi(q_0)\in Q_F\}$ and, for $i\in I$,
$g(i)=\{(i,f)\mid f\colon V\to\Phi\}$.

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

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 $V=\{\sigma_1, \ldots, \sigma_n\}$ 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 $(1\leqq j\leqq n)$, 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 dΓK-iteration grammars. We may obviously assume that all V_{ij} are disjoint. We now construct a dΓK-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 $1\leqq j\leqq n$, and $\mu_i(\sigma)=\{F\}$ for $\sigma\notin V$. For each $k\in I_{ij}$, μ_k is defined by

$\mu_k(\sigma)=v_k(\sigma)$ for $\sigma\in V_{ij}$ (where $U_{ij}=\{v_k\mid k\in I_{ij}\}$),
$\mu_k(\sigma)=\{\sigma\}$ for $\sigma\notin V_{ij}$.

The d-substitution μ_t is defined by

$\mu_t(\sigma)=\{\phi_{ij}^{-1}(\sigma)\}$ if $\sigma\in T_{ij}$ for some $i\in I$ and $1\leqq j\leqq n$,
$\mu_t(\sigma)=\{F\}$ otherwise.

The new control language M_0 is as follows. Let $I=\{i_1, \ldots, i_p\}$. Then $M_0=(M_1^*\ldots M_p^*)^*=(M_1\cup\cdots\cup M_p)^*$, where for $1\leqq s\leqq p$ $M_s=i\cdot M_{i1}\ldots M_{in}\cdot t$ (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. □

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 $\eta(K)\subseteq K_0$. □

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

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

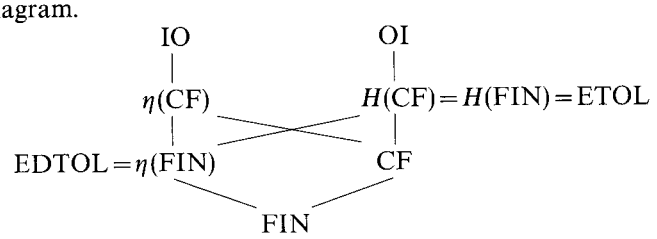
Proof. By Theorem 4.4, $\eta(\text{ONE}\cup\{\emptyset\})=\text{EDTOL}$ is a full dhyper-QAFL. Let K be a full dhyper-QAFL. Hence K is a pseudoid and $\eta(K)\subseteq K$. Then ONE$\cup\{\emptyset\}\subseteq K$, and so $\eta(\text{ONE}\cup\{\emptyset\})\subseteq\eta(K)\subseteq K$. □

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 $F(G)=\{\omega\in V^*\mid S\overset{*}{\Rightarrow}\omega\}$ 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 $\eta(\mathrm{CF})$ 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 $\eta(\mathrm{CF})$ and IO. Clearly, $\mathrm{CF} \subseteq \mathrm{IO}$ implies $\eta(\mathrm{CF}) \subseteq \eta(\mathrm{IO})$. But we just showed that IO is closed under iterated d-substitution, i.e. $\eta(\mathrm{IO}) \subseteq \mathrm{IO}$. Hence $\eta(\mathrm{CF}) \subseteq \mathrm{IO}$.

Incomparability of EDTOL and CF was proved in [6]. There is a language in ETOL—IO (cf. [11]). There is also a language in $\eta(\mathrm{CF})$—OI: if $L \in \mathrm{CF}$—EDTOL, then $\{w\#w\#w \mid w \in L\}$ is obviously in $\eta(\mathrm{CF})$, but not in OI [10]. The proper inclusion of ETOL in OI was shown in [7]. Finally, the proper inclusion of $\eta(\mathrm{CF})$ 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 $\eta(\mathrm{CF})$. 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 $H_1(\mathrm{REG})$ and $H_1(\mathrm{FIN})$, 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 $\eta(\mathrm{CF})$ 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, $\eta(\mathrm{CF})$ 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 $\sigma_1\sigma_2 \overset{*}{\Rightarrow} \omega_1\omega_2$, then $\sigma_1\sigma_2\sigma_2\sigma_1 \overset{*}{\Rightarrow} \omega_1\omega_2\omega_2\omega_1$). Let, in these lemmas, $G=(V, T, I, U, S)$ be a d-iteration grammar.

5.1. Lemma. *Let $\sigma_1, \ldots, \sigma_k \in V (k \geqq 1)$, $\omega \in V^*$ and $\pi \in I^*$. If $\sigma_1 \ldots \sigma_k \overset{\pi}{\Rightarrow} \omega$, then there are $\omega_1, \ldots, \omega_k \in V^*$ such that $\omega = \omega_1 \ldots \omega_k$, $\sigma_j \overset{\pi}{\Rightarrow} \omega_j$ for $1 \leqq j \leqq k$, and if $\sigma_i = \sigma_j$ then $\omega_i = \omega_j$.* □

5.2. Lemma. *Let $\sigma_1, \ldots, \sigma_k \in V$, $\omega_1, \ldots, \omega_k \in V^*$ $(k \geqq 1)$ and $\pi \in I^*$. If $\sigma_1 \ldots \sigma_k \overset{\pi}{\Rightarrow} \omega_1 \ldots \omega_k$ and $\sigma_j \overset{\pi}{\Rightarrow} \omega_j$ for $1 \leqq j \leqq k$, then $\sigma_{j_1}\sigma_{j_2} \ldots \sigma_{j_n} \overset{\pi}{\Rightarrow} \omega_{j_1}\omega_{j_2} \ldots \omega_{j_n}$ for all j_s $(1 \leqq s \leqq n)$ such that $1 \leqq j_s \leqq k$.* □

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 $n \geqq 1$ and all $u, v, x_0, x_1, \ldots, x_{2n} \in T^*$, if $x_0 u^k x_1 v^k x_2 u^k x_3 v^k x_4 \ldots x_{2n-2} u^k x_{2n-1} v^k x_{2n} \in L$ for all $k \geqq 0$, then $u = v = \lambda$. □

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 $f\colon T_1^* \to T_2^*$ and $g\colon T_1^* \to T_3^*$ 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 $\phi_1 = x_0 u x_1 v x_2 \ldots x_{2n-2} u x_{2n-1} v x_{2n}$ and $\phi_2 = x_0 u^2 x_1 v^2 x_2 \ldots$ are both in L, and that $uv \neq \lambda$. Then clearly, u and v cannot contain $\#$. Thus uv 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.) □

The next theorem is the main result of this section.

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

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 $L(G) \in \mathrm{EDTOL} = \eta(\mathrm{FIN})$. 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 $|z| \geqq q$, 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 $k \geqq 0$ ($|z|$ 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 $\geqq q$ 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 $L(G) \in \eta(\mathrm{FIN}) = \mathrm{EDTOL}$.

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

$(1 \leqq j \leqq r+1)$, $z_j \overset{\pi}{\Rightarrow} y_j$ $(j=1,2,3)$, $s \overset{\pi}{\Rightarrow} u$ and $t \overset{\pi}{\Rightarrow} v$. We are now going to change the derivation $S \overset{*}{\Rightarrow} w$ 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:

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

(where the derivation according to π is justified by application of Lemma 5.2: several copies of the derivations $s \overset{\pi}{\Rightarrow} u$ and $t \overset{\pi}{\Rightarrow} v$, which were already present, are added to the original derivation). Hence, for all $k \geqq 0$, $w(k) \in L(G)$. 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 $st \neq \lambda$. □

As a consequence of this theorem we can immediately show that η(CF) is not closed under "colouring".

5.6. Corollary. *η(CF) is a full dhyper-QAFL not closed under deterministic sequential machine mappings.*

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

Finally we prove the proper inclusion of η(CF) in IO (cf. the end of Section 4).

5.7. Corollary. *η(CF) is properly included in IO.*

Proof. In [11] it is shown that the language $L = \{(b^k a)^m b^k \mid k \geqq 1, m = 2^k - 1\}$ is in IO—EDTOL. We shall show that L has property P. This implies, by Theorem 5.5, that $L \notin \eta$(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 $1 \leqq n|uv|_a \leqq 2^k - 1$. Hence $2^k \leqq |\phi_2|_a \leqq 2^k - 1 + 2^k - 1 = 2^{k+1} - 2$. 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. □

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 indispensible. 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 OI. 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

Supplement to Chapters 2 and 3

Three Notes on Controlled Hyper-Algebraic and Dhyper-Algebraic Extensions

Reprinted from *TW-memorandum No. 192 (November 1977)*

Abstract

1. Regular control does not increase the generating power of 1-restricted [d]K-iteration grammars provided that K $\supseteq$ SYMBOL, and K is closed under isomorphism and under union with SYMBOL-languages.

2. Let Γ be a prequasoid closed under the regular operations. If K is a prequasoid [pseudoid], then $H(\Gamma) \subseteq H(\Gamma,K)$ [$\eta(\Gamma) \subseteq \eta(\Gamma,K)$]. In particular we have $H(\Gamma) \subseteq (\Gamma)$ETOL and $\eta(\Gamma) \subseteq (\Gamma)$EDTOL.

3. Under weak conditions on Γ and K, the decidability of the emptiness problem for Γ and K implies the decidability of the emptiness problem and the membership problem for the families $\eta(\Gamma,K)$ and $\eta(K)$.

1. Regularly Controlled 1-Restricted Hyper-Algebraic and Dhyper-Algebraic Extensions.

In [1,3] we showed that regular control does not increase the generating capacity of [d]hyper-algebraic extensions. In the proof the number of substitutions is however increased with 1 [1,3] and since we can only reduce the number of substitutions to 2 in general [1, 3], the argument cannot be applied in the 1-restricted case, i.e. to [d-]iteration grammars containing only one [d-]substitution.

In this note we show that for 1-restricted [d]hyper-algebraic extensions regular control does also not provide any additional generating power; thus yielding a similar result as in the unrestricted case.

Remember that a family K of languages is called α-simple [2] if K includes SYMBOL and if K is closed under isomorphism ("renaming of symbols") and under union with SYMBOL-languages. (For all unexplained terminology we refer to [1, 2, 3] and the references mentioned there).

Theorem. If K is α-simple, then

(i) $H_1(REG,K) = H_1(K)$

(ii) $\eta_1(REG,K) = \eta_1(K)$.

Proof: (i) Obviously $H_1(K) \subseteq H_1(REG,K)$.

Conversely, let $G = (V,\Sigma,\tau,M,S)$ be a (REG,K)-iteration grammar with a single K-substitution τ. Let $(Q,\{\tau\},\delta,q_0,Q_F)$ be a complete deterministic finite state acceptor for M, where Q is the set of states, $\{\tau\}$ is the input alphabet, $\delta : Q \times \{\tau\} \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $Q_F \subseteq Q$ is the set of final states.

Consider the K-iteration grammar $H = (V_0,\Sigma,\tau_0,S_0)$, where $V_0 = V \times Q \cup \Sigma \cup \{F\}$, $S_0 = [S,q_0]$, and F is a rejection symbol.

Distinguish the following cases.

Case 1: M is finite.

We may assume that each state in Q is non-recurrent (i.e. it is impossible to visit a state more than one time).

Then we define

$$\tau_0([\alpha,q]) = \{[\alpha_1,q'] \ldots [\alpha_n,q'] \mid n \geq 0,\ \alpha_1 \ldots \alpha_n \in \tau(\alpha);\ \delta(q,\tau) = q';\ q \in Q\} \cup T(\alpha,q) \cup \{F\} \quad \text{if } \alpha \in V, \text{ with}$$

$$T(\alpha,q) = \begin{cases} \{\alpha\} & \text{if } q \in Q_F \text{ and } \alpha \in \Sigma \\ \emptyset & \text{otherwise} \end{cases}$$

$$\tau_0(\alpha) = \{F\} \quad \text{if } \alpha \in \Sigma \cup \{F\}.$$

Obviously, we have $L(H) = L(G)$, and hence $H_1(REG,K) \subseteq H_1(K)$.

Case 2: M is infinite.

In this case the substitution τ_0 is defined as follows:

$$\tau_0([\alpha,q]) = \{[\alpha_1,q'] \ldots [\alpha_n,q'] \mid n \geq 0,\ \alpha_1 \ldots \alpha_n \in \tau(\alpha);\ \delta(q,\tau) = q'\} \cup T(\alpha,q) \cup \{F\} \quad \text{if } \alpha \in V,\ q \in Q$$

where $T(\alpha,q)$ is as in Case 1.

$$\tau_0(\alpha) = \{F\} \quad \text{if } \alpha \in \Sigma \cup \{F\}$$

It is easy to prove that $L(H) \cap \Sigma^+ = L(G) \cap \Sigma^+$. As usual the only (possible) difficulty is caused by the empty word.

Assume $\lambda \in L(G)$, i.e. there exists $n \geq 1$ with $\lambda \in \tau^n(S)$ and $\tau^n \in M$. By the construction of H, we have $\lambda \in \tau_0^n(S_0)$. On the other hand we have to show that $\lambda \in L(H)$ only if $\lambda \in L(G)$.

Suppose $\lambda \in L(H)$, then two possibilities occur.

(α) There exists a derivation $S_0 \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_n = \lambda$ according to H such that $w_i \neq \lambda$ $(1 \leq i \leq n-1)$ and $\tau^n \in M$. From the construction of H it follows that $\lambda \in L(G)$.

(β) There exists a derivation $S_0 \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_n = \lambda$ according to H such that $w_i \neq \lambda$ $(1 \leq i \leq n-1)$ and $\tau^n \notin M$. Since M is infinite, there is a controlword $\tau^p \in M$ with $p > n$. Then $\lambda \in \tau^p(S)$ and hence $\lambda \in L(G)$ as

$\tau^{p-n}(\lambda) = \{\lambda\}$. (Note that in the subderivation $w_n \Longrightarrow \ldots \Longrightarrow w_p = \lambda$, $w_i = \lambda$ $(n \leq i \leq p)$ no state from Q is attached to λ and so we lost the remaining part of the controlword. This loss is however quite immaterial since $\tau_0^{p-n}(\lambda)$ can only yield λ).

So we have $L(H) = L(G)$ and consequently $H_1(REG,K) \subseteq H_1(K)$.

(ii) Exactly the same construction can be applied in the deterministic case. □

2. A Remark on Controlled Hyper-Algebraic and Dhyper-Algebraic Extensions.

In [1] we showed that under weak assumptions on the families Γ and K the Γ-controlled hyper-algebraic extension $H(\Gamma,K)$ of K is a full hyper-AFL including the families K, Γ and $H(K)$. Similarly, in the deterministic case [3] we have that under weak conditions on Γ and K the Γ-controlled dhyper-algebraic extension $\eta(\Gamma,K)$ of K is a full dhyper-QAFL including the families K, Γ and $\eta(K)$.

We now prove that under an additional assumption on Γ the family $H(\Gamma,K)$ [$\eta(\Gamma,K)$] also includes $H(\Gamma)$ [respectively $\eta(\Gamma)$]. (This result has been inspired by a remark in [4]).

Remember that a full [FIN,REG]-structure [2] is a quasoid closed under the regular operations (union, concatenation and Kleene *).

Theorem. Let Γ be a full [FIN,REG]-structure.

(i) If K is a prequasoid, then $H(\Gamma) \subseteq H(\Gamma,K)$

(ii) If K is a pseudoid, then $\eta(\Gamma) \subseteq \eta(\Gamma,K)$.

Proof: Let $M \subseteq \Sigma^*$ be a language in Γ and let $a,b \notin \Sigma$. Both {a} and {b} are in Γ, because Γ is a quasoid and therefore $REG \subseteq \Gamma$. Since Γ is closed under concatenation we have $aMb \in \Gamma$, i.e. Γ is closed under full marking.

(i) The main result in [1] implies that $H(\Gamma,K)$ is a full hyper-AFL including Γ. Since Γ is a prequasoid, $H(\Gamma)$ is the smallest full hyper-AFL including Γ[1,2]. Hence $H(\Gamma) \subseteq H(\Gamma,K)$.

(ii) From [3] it follows that $\eta(\Gamma,K)$ is a full dhyper-QAFL. Similar to Lemma 4.1 in [1] it is easy to show that $\eta(\Gamma,K)$ includes Γ. As Γ is a pseudoid, $\eta(\Gamma)$ is the smallest full dhyper-QAFL including Γ [3]. Hence $\eta(\Gamma) \subseteq \eta(\Gamma,K)$. □

Corollary [4]. Let Γ be a full [FIN, REG]-structure. Then

$$H(\Gamma) \subseteq H(\Gamma,FIN) = (\Gamma)ETOL$$

$$\eta(\Gamma) \subseteq \eta(\Gamma,ONE) = (\Gamma)EDTOL$$

3. Decision Problems for the Families $\eta(\Gamma,K)$ and $\eta(K)$.

In the following we restrict our attention to effective closure properties only (cf. [1] Section 7).

Since [1] Lemma 7.1 (i.e. [5] Theorem 1) can easily be proved for dK-iteration grammars [3] too, we obtain by an argument almost identical to the non-deterministic case (cf. [1] Section 7) the following decidability results.

Theorem.

(i) Let Γ and K be closed under intersection with regular languages and let the emptiness problem be decidable in Γ and K. Then the emptiness problem is decidable for languages in $\eta(\Gamma,K)$.

(ii) Let K be a pseudoid and let Γ be closed under full marking and intersection with regular languages. If the emptiness problem is decidable in Γ and K, then the membership problem is decidable in $\eta(\Gamma,K)$.

Corollary.

(i) If K is closed under intersection with regular languages and if the emptiness problem is decidable for languages in K, then the emptiness problem is also decidable in $\eta(K)$.

(ii) Let K be a pseudoid and let the emptiness problem be decidable in K. Then the membership problem is decidable in $\eta(K)$.

References

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

2. P.R.J. Asveld, Extensions of language families and canonical forms for full AFL-structures, TW-memo No. 167, Twente University of Technology, Enschede, The Netherlands.

3. P.R.J. Asveld, J. Engelfriet, Iterated deterministic substitution, Acta Informatica 8 (1977) 285-302.

4. W.J. Erni, Auxiliary pushdown acceptors and regulated rewriting systems, Report 59, Institut für angewandte Informatik und formale Beschreibungsverfahren, Universität Karlsruhe (1977).

5. D. Wood, A note on Lindenmayer systems, Szilard languages, spectra, and equivalence, Internat. J. Comp. Inform. Sci. 4 (1975) 53-62.

Chapter 4

Extended Linear Macro Grammars, Iteration Grammars, and Register Programs

with *Joost Engelfriet*

Reprinted from *TW-memorandum No. 209 (March 1978)*

Abstract

Extended macro grammars (of the linear basic type only) are introduced as a generalization of those in [5], and it is shown that they have the same language generating power as (parallel) iteration grammars. In particular the IO and OI versions of extended macro grammars correspond to the deterministic and the (usual) nondeterministic iteration grammars respectively. Hence iterated substitution can be formulated using extended macro grammars.

A nondeterministic register program without tests may be viewed as a macro grammar. IO-extension of this macro grammar corresponds to the use of nonrecursive function procedures in the register program. OI-extended macro grammars correspond to register programs which compute on sets. Hence these features of register programs can be investigated by means of (extended) parallel rewriting systems.

1. Introduction

In [5, 1] a relationship has been established between certain classes of parallel rewriting systems and macro grammars. In particular EDTOL systems [23] have the same language generating power as linear basic (lb) macro grammars ([12]; a macro grammar is linear basic if the right-hand side of each rule contains at most one nonterminal; an example is the macro grammar with rules $S \to A(\lambda,\lambda)$, $A(x,y) \to A(xa,yb)$, $A(x,y) \to xyx$, generating the language $\{a^n b^n a^n \mid n \geq 0\}$). Whereas macro grammars in general may be viewed as (and were suggested by) nondeterministic recursive procedures, the restricted class of lb macro grammars is also well suited to model a simple type of register program. As an example, the above grammar corresponds to the register program

start(x:=λ, y:=λ); repeat(x:=xa, y:=yb); halt(xyx),

where the repeat statement should be repeated any number of times. Thus EDTOL is the family of languages generated by nondeterministic register programs (without tests) which keep strings in their registers and use concatenation as basic operation (and as basic constants the symbols of the alphabet and the empty string λ). Since these register programs are such a natural language generating device, the results of [5, 1] increased our interest in the family of EDTOL languages. In [5] it was also shown that the ETOL languages [23] can be generated by register programs with an additional nondeterministic feature: they can keep finite languages in their registers and use concatenation and union (of finite languages) as basic operations. The corresponding type of macro grammar, in which finite languages can occur in the arguments of a nonterminal, was called "extended" linear basic macro grammar (It is not a macro grammar as such, but it can easily be simulated by one). As an example (adapted from [12 , Example 1.2.5]) consider the extended lb macro grammar with rules $S \to A(\{a,b\})$, $A(x) \to A(\{xa,xb\})$, $A(x) \to B(\lambda,x)$, $B(y,x) \to B(yxc,x)$ and $B(y,x) \to yx$, where $\{xa,xb\}$ should be

interpreted as xa ∪ xb, thus $A(\{a,b\}) \Longrightarrow A(\{aa,ab,ba,bb\})$. This grammar generates the "equal length" language $\{w_1cw_2c\ldots cw_n \mid n \geq 1, w_i \in \{a,b\}^*$ and $|w_i| = m$ for some $m \geq 1$ and all $i\}$.

In this paper we study extensions of linear basic macro grammars in a more general setting.

We consider lb macro grammars with rules of the form $A(x_1,\ldots,x_n) \rightarrow B(\psi_1,\ldots,\psi_m)$ or $A(x_1,\ldots,x_n) \rightarrow \psi$, where A,B are nonterminals and $\psi_1,\ldots,\psi_m$ $(m \geq 0)$ and ψ are languages over the terminal symbols and the variables $x_1,\ldots,x_n$ $(n \geq 0)$, these languages being taken from a given family K of languages. Thus each rule represents a countable number of alternatives, restricted by the family K. This extension mechanism is well known from Extended BNF which can be viewed as a context-free grammar in which the right-hand sides of rules are regular languages (cf. [28] for extended context-free grammars in general), and from the theory of K-iteration grammars [27, 26, 3, 4] which are extended EDTOL systems with right-hand sides taken from a family K of languages.

There are two, essentially different, ways of defining the working of such extended lb macro grammars: the OI-extension and the IO-extension. In the OI-extension (of which the extension of [5] is the particular case that K is the family of finite languages) the countably many alternatives which are present in a rule, are handled simultaneously by keeping languages in the registers (i.e. argument positions of the nonterminals) using concatenation and countable union as basic operations (interpreting each language ψ as the countable union of its strings). In the IO-extension one possible choice is made from the countably many alternatives, each time the rule is applied; thus the registers contain strings and the rule $A(x_1,\ldots,x_n) \rightarrow B(\psi_1,\ldots,\psi_m)$ is viewed as an abbreviation of the countably many rules $A(x_1,\ldots,x_n) \rightarrow B(w_1,\ldots,w_m)$ with $w_i \in \psi_i$. When the grammar for the "equal length" language mentioned above, is viewed in the IO-way, it generates the language $\{(wc)^n w \mid n \geq 0, w \in \{a,b\}^*\}$. As another example (with K the family of regular

languages) consider the extended lb macro grammar with rules $S \to A(d^*a \cup d^*b)$ $A(x) \to A(xd^*a \cup xd^*b)$, $A(x) \to B(\lambda,xd^*)$, $B(y,x) \to B(yxc,x)$ and $B(x) \to yx$, and let h be the homomorphism with $h(d) = \lambda$ and the identity on a,b and c. In the OI-way this grammar generates $h^{-1}(L)$, where L is the equal length language, whereas in the IO-way it generates $\{(wc)^n w \mid n \geq 0, w \in \{a,b,d\}^*\}$.

The main two results of this paper are described in the following paragraphs (1) and (2). The feeling that these results could be obtained also constituted our motivation to study extensions of linear basic macro grammars.

(1) The register programs mentioned above are a natural device to define subsets of an arbitrary algebra A by putting expressions (trees) rather than strings in the right-hand sides of assignments. Linear basic macro grammars (and their extended versions) can be used to investigate these subsets, since the "Mezei and Wright like" result holds that each such subset is the homomorphic image of the corresponding linear basic (tree) language (In [20] a similar result was obtained for context-free grammars and equational subsets of an algebra). In this context, extension means that a family K of subsets of A is given in advance and that the "extended register programs" define all subsets which are computable from the elements of K. OI-extension corresponds roughly to using the register programs to compute in the subset algebra of A. IO-extension corresponds to the use of (nondeterministic) nonrecursive function procedures in the right-hand side of assignments. Thus both ways of extending lb macro grammars give rise to a natural way of extending register programs.

(2) Extended lb macro grammars provide an alternative description of iteration grammars (i.e. extended EDTOL systems). The OI-extended lb macro grammars are equivalent to the usual iteration grammars (for each given K satisfying a few closure properties), as one could expect from the results of [5]. The IO-extended lb macro grammars turn out to be equivalent to the "deterministic" iteration grammars introduced in [4], in which all occurrences of a given symbol in a sentential form have to be rewritten by the same string.

Thus both ways of extending lb macro grammars naturally correspond to both ways of extending EDTOL systems to iteration grammars. This correspondence also holds in the particular case of trees (cf. [5, 1] for a definition of lb tree grammars and EDTOL tree systems, and their equivalence).

These two results (1) and (2) show that there is a close relationship between (parallel) iteration grammars and nondeterministic register programs with the extended lb macro grammars as an intermediate stage (The relation between formal language theory and the theory of programs in general is well known; cf. e.g. [7, 13, 15, 21]). This relationship implies that
(a) closure properties of families of iteration languages can be proved by the method of "register programming" which is often more familiar than the construction of iteration grammars, and
(b) iteration grammars provide a simple tool to investigate register programs of the type described above.

Throughout the paper we also study the addition of control to all devices considered. Controlled extended lb macro grammars (defined in the obvious way) are equivalent to controlled iteration grammars (considered e.g. in [3, 4]), but the control words have to be reversed. This is in accordance with the fact that lb macro grammars and iteration grammars can be viewed as computing each others fixed point (of the set of equations corresponding to the grammar): fixed points of macro grammars can be computed by iterating substitution [5, 9] and fixed points of iteration grammars are computed by recurrence systems [16], i.e. by macro grammars. It should be clear that a control structure can also be added to register programs, preserving the results in (1). Thus, as an example, context-free controlled EDTOL systems correspond to context-free controlled register programs, which may be viewed as recursive register programs (with global registers and parameterless procedures).

This paper consists of four sections of which the present introductory

section is the first one. In Section 2 we introduce the basic notion of (controlled) extended lb macro grammar. By representing the languages on the argument positions by special nonterminals we can uniformly define an extended lb macro grammar as a (non-linear) macro grammar (with countably many productions), such that the OI and IO derivations of this grammar define the OI-extension and IO-extension respectively. We also establish two lemmas on controlled extended lb macro grammars which facilitate our proofs in Section 3.

In Section 3 we recall the concept of iteration grammar and we prove one of our main results (cf. (2) above), viz. the equivalence between (controlled) IO-extended lb grammars and (controlled) deterministic iteration grammars, and similarly between (controlled) OI-extended lb grammars and (controlled) nondeterministic iteration grammars. As an application we show the (known) fact that the family of IO [OI] macro languages is closed under iterated deterministic [nondeterministic] substitution.

In Section 4 we study register programs with expressions (trees) in the right-hand side of assignments, and we establish the other main result (cf. (1) above): their generating power in an algebra exactly equals those of extended lb tree grammars. By a "tree analogue" of the results in Section 3 we also obtain their equivalence with iteration tree grammars in the IO/deterministic, as well as in the OI/nondeterministic case. Section 4 is concluded with a few examples of how to take advantage of this relationship between register programs and iteration grammars (cf. (a) and (b) above). In particular, the context-free controlled EDTOL tree languages are compared with the IO and OI macro tree languages (all these may be viewed as recursive program schemes).

For all unexplained notation and terminology on formal languages and parallel rewriting (in the sense of L systems) the reader is referred to [18, 25] and [17].

In particular we use CF, REG, FIN, ONE and SYMBOL to denote the families of context-free languages, regular languages, finite languages, singleton languages (i.e. languages containing precisely one word), and singletons of length one respectively.

2. Extended Linear Basic Grammars

In this section we first recall some basic concepts and terminology on macro grammars. Then we introduce the principal definition of (controlled) extended lb grammar, and finally we establish two helpful lemmas (Lemmas 2.9 and 2.10).

Macro grammars have been introduced in [12] as a generalization of context-free grammars in order to model certain non-context-free features in the syntax of programming languages. They essentially differ from context-free grammars in having a ranked alphabet of nonterminals and consequently macro grammars are actually rewriting systems on terms rather than on (simple) strings. The left-hand side of each production consists of a nonterminal symbol (of rank n) provided with n formal parameters (or variables) $x_1,\ldots,x_n$ whereas the right-hand side is a term over the union of $\{x_1,\ldots,x_n\}$ and the alphabet of the grammar. So each production may be considered as a macro definition and it is applied by expanding that macro with either outermost calls first (OI or outside-in mode of derivation) or innermost calls first (IO or inside-out mode of derivation) yielding the family of OI and IO macro languages respectively.

In order to be more precise we need the concept of term over a ranked alphabet. Recall that a ranked alphabet Δ is a finite set of symbols such that with each symbol A in Δ a unique nonnegative integer (the rank of A) is associated. For each $i \geq 0$ we define Δ_i to be the subalphabet of Δ consisting

of all symbols of rank i (Note that $\Delta_i \cap \Delta_j = \emptyset$ for $i \neq j$). Let PC denote the alphabet of punctuation characters, i.e. PC consists of the left parenthesis, the right parenthesis and the comma symbol. The set of terms over Δ is the least set of strings over $\Delta \cup PC$ satisfying; (i) Each element of $\Delta_0 \cup \{\lambda\}$ is a term (λ denotes the empty word); (ii) If t_1 and t_2 are terms, then $t_1 t_2$ is a term; (iii) If $A \in \Delta_m$ and $t_1,\ldots,t_m$ are terms ($m \geq 1$), then $A(t_1,\ldots,t_m)$ is a term.

A macro grammar (Φ,Σ,X,P,S) consists of

(1) a ranked alphabet Φ of nonterminals

(2) a terminal alphabet Σ

(3) a finite set of variables X

(Each terminal and variable has rank zero; Φ, Σ and X are assumed to be disjoint)

(4) an initial nonterminal S in Φ_0, i.e. S has rank zero

(5) a finite set P of productions or rules, each of the form $A(x_1,\ldots,x_n) \rightarrow t$, where A is in Φ_n, $x_1,\ldots,x_n$ are mutually distinct elements of X, and t is a term over $\Sigma \cup (\Phi - \{S\}) \cup \{x_1,\ldots,x_n\}$.

In order to define modes of derivation for a macro grammar we need the concept of subterm and of top level. If t_1 is a term over Δ, then t_2 is a subterm of t_1, whenever t_2 is a term over Δ and t_2 is a substring of t_1. A subterm t_2 of t_1 occurs at top level in t_1, if t_2 does not appear within the argument list of some symbol of Δ which occurs in t_1.

We now recall the formal definitions of OI (outside-in) and IO (inside-out) mode of derivation [12]. Let $G = (\Phi,\Sigma,X,P,S)$ be a macro grammar and let σ and τ be terms over $\Phi \cup \Sigma$. Then we write $\sigma \underset{OI}{\Longrightarrow} \tau$ if

- σ contains a subterm (over $\Phi \cup \Sigma$) of the form $A(\xi_1,\ldots,\xi_n)$ occurring at top level, where A is in Φ_n and $\xi_1,\ldots,\xi_n$ are terms over $\Phi \cup \Sigma$.
- $A(x_1,\ldots,x_n) \rightarrow t$ is a production in P.

- τ is obtained from σ by replacing that (top level) occurrence of $A(\xi_1,\ldots,\xi_n)$ by t', where t' is the result of substituting the terms $\xi_1,\ldots,\xi_n$ for $x_1,\ldots,x_n$ in t respectively.

We write $\sigma \underset{IO}{\Longrightarrow} \tau$ if

- σ contains a subterm (over $\Phi \cup \Sigma$) of the form $A(\xi_1,\ldots,\xi_n)$ where A is in Φ_n and $\xi_1,\ldots,\xi_n \in \Sigma^*$ (i.e. $\xi_1,\ldots,\xi_n$ are terms over Σ).
- $A(x_1,\ldots,x_n) \to t$ is a production in P.
- τ results from replacing that occurrence of $A(\xi_1,\ldots,\xi_n)$ in σ by t', where t' is obtained by substituting the words $\xi_1,\ldots,\xi_n$ for the corresponding variables $x_1,\ldots,x_n$ in t.

As usual $\underset{OI}{\Longrightarrow}^*$ [$\underset{IO}{\Longrightarrow}^*$] denotes the reflexive and transitive closure of the binary relation $\underset{OI}{\Longrightarrow}$ [$\underset{IO}{\Longrightarrow}$].

Let m be a mode of derivation, i.e. m equals either OI or IO. An m-macro grammar is a macro grammar provided with the mode m. The language generated by an m-macro grammar $G = (\Phi,\Sigma,X,P,S)$ is defined as usual, viz. $L_m(G) = \{w \in \Sigma^* \mid S \underset{m}{\Longrightarrow}^* w\}$. The family of languages generated by OI and IO macro grammars are respectively denoted by OI and IO.

We now turn to the class of linear basic grammars. A linear basic grammar (abbreviated by lb grammar) [12] is a macro grammar in which the right-hand side of each rule contains at most one occurrence of a nonterminal symbol. So each sentential form obtained by means of an lb grammar also contains at most one occurrence of a symbol of Φ and consequently, the OI and IO mode of derivation coincide for this class of macro grammars. Hence the family of languages generated by lb grammars is included in OI $\cap$ IO.

It may be assumed without loss of generality [12], that each production in an lb grammar has one of the forms

(*)
$$A(x_1,\ldots,x_n) \to B(w_1,\ldots,w_k), \quad \text{or}$$
$$A(x_1,\ldots,x_n) \to w_0$$

where $w_0,w_1,\ldots,w_k$ are words over $\Sigma \cup \{x_1,\ldots,x_n\}$.

During the last few years several generalizations of lb grammars have been considered. Downey [5] investigated "extended lb grammars", i.e. lb grammars with productions of the form (*) in which each single word $w_0,\ldots,w_k$ is replaced by a finite language over $\Sigma \cup \{x_1,\ldots,x_n\}$. In [10] these grammars have been generalized further by allowing regular languages over $\Sigma \cup \{x_1,\ldots,x_n\}$ in (*).

We now introduce the general case that each word w_i $(0 \leq i \leq k)$ in (*) is replaced by a language from an arbitrary family K. Instead of simply putting L_i for w_i in (*) with $L_i \in K$, we follow a more subtle approach which enables us to distinguish OI and IO derivations for these grammars in a natural way. So we replace each w_i $(0 \leq i \leq k)$ in (*) by a language name $\psi_i(x_1,\ldots,x_n)$, i.e. an element from a special ranked alphabet. Then we extend the set of productions P with unique productions of the form $\psi_i(x_1,\ldots,x_n) \to L_i$, where $L_i \subseteq (\Sigma \cup \{x_1,\ldots,x_n\})^*$ is a language in the family K. This leads to the following more formal definition.

Definition 2.1. Let K be a family of languages. An extended linear basic K-grammar or K-elb grammar is a 6-tuple (Φ,Ψ,Σ,X,P,S) where

- Φ is a ranked alphabet of nonterminals
- Ψ is a ranked alphabet of language names
- Σ is a terminal alphabet
- X is a finite set of variables

(Variables and terminals have rank zero. The sets Φ,Ψ,Σ and X are assumed to be mutually disjoint)

- $S \in \Phi_0$ is the initial nonterminal
- P is a finite set of productions or rules each having one of the following three forms

(i) $A(x_1,\ldots,x_n) \to B(\psi_1(\underline{x}),\ldots,\psi_k(\underline{x}))$, where $A \in \Phi_n$, $B \in \Phi_k - \{S\}$, $(\underline{x}) = (x_1,\ldots,x_n)$, and $x_1,\ldots,x_n$ are mutually distinct elements of X $(k,n \geq 0)$. As usual we write "A" instead of "A( )" in case that n=0, and similarly for members of Ψ.

(ii) $A(x_1,\ldots,x_n) \to \psi(x_1,\ldots,x_n)$, where $A \in \Phi_n - \{S\}$, $\psi \in \Psi_n$, and $x_1,\ldots,x_n$ are mutually distinct elements of X $(n \geq 0)$.

(iii) $\psi(x_1,\ldots,x_n) \to L$, where $\psi \in \Psi_n$ $(n \geq 0)$, $x_1,\ldots,x_n$ are mutually distinct members of X, and $L \subseteq (\Sigma \cup \{x_1,\ldots,x_n\})^*$ is a language in K.

We require moreover that there is exactly one production of the form (iii) in P for each ψ in Ψ. □

Each K-elb grammar $G = (\Phi,\Psi,\Sigma,X,P,S)$ can be viewed as a macro grammar G' with a countable (rather than a finite) number of productions in the following way. Let $G' = (\Phi \cup \Psi,\Sigma,X,P',S)$ where the countable set P' is defined as follows.

(1) Each rule in P of the form 2.1(i) or 2.1(ii) is also in P'.

(2) For each production $\psi(x_1,\ldots,x_n) \to L$ of the form 2.1(iii) in P, P' contains the (countable number of) productions

$\psi(x_1,\ldots,x_n) \to w$ for each w in L.

For m equal to either OI or IO, we use the countable macro grammar G' to provide G with a mode of derivation (Clearly the definitions of OI and IO mode of derivation do not depend on the finiteness of the set of productions). So by definition a K-elb grammar G is an m-extended linear basic K-grammar or (m,K)-elb grammar (i.e. a K-elb grammar provided with the mode m) if G' is a m-macro grammar. The language $L_m(G)$ generated by G is defined by $L_m(G) = L_m(G')$. The family of languages generated by (m,K)-elb grammars is denoted by $LB_m(K)$. Note that ONE-elb grammars correspond directly to linear basic grammars, and so $LB_m(ONE)$ is the family of languages generated by lb grammars.

The following two remarks concern the structure of derivations according to K-elb grammars.

Remark 2.2. Let $G = (\Phi,\Psi,\Sigma,X,P,S)$ be an (OI,K)-elb grammar. Then each derivation according to G has the form

$$S \underset{OI}{\Longrightarrow} t_1 \underset{OI}{\Longrightarrow}^* t_f \underset{OI}{\Longrightarrow}^* t_n \qquad (1 \leq f < n),$$

where t_{f-1} is obtained from S by only applying rules of the form 2.1(i), t_{f-1} yields t_f by a single application of a production of the form 2.1(ii) and in the subderivation $t_f \underset{OI}{\Longrightarrow}^* t_n$ only rules of the form 2.1(iii) are used.

Consequently $S,t_1,\ldots,t_{f-1}$ contain exactly one nonterminal, whereas $t_f,t_{f+1},\ldots,t_n$ do not contain any nonterminal symbol. □

Remark 2.3. Let $G = (\Phi,\Psi,\Sigma,X,P,S)$ be an (IO,K)-elb grammar. Then each derivation according to G has the form

$$S \underset{IO}{\Longrightarrow} t_1 \underset{IO}{\Longrightarrow}^* t'_1 \underset{IO}{\Longrightarrow} t_2 \underset{IO}{\Longrightarrow}^* t'_2 \underset{IO}{\Longrightarrow} \ldots \underset{IO}{\Longrightarrow}^* t'_{n-1} \underset{IO}{\Longrightarrow} t_n \underset{IO}{\Longrightarrow}^* t'_n, \quad (n > 1)$$

where t_p is obtained from t'_{p-1} $(1 \leq p < n$; assume $t'_0 = S)$ by a rule of the form 2.1(i), t'_{n-1} yields t_n by an application of a production of the form 2.1(ii), and in the subderivations $t_p \underset{IO}{\Longrightarrow}^* t'_p$ $(1 \leq p \leq n)$ only rules of the form 2.1(iii) are used.

Consequently, $t_1,\ldots,t_{n-1}$, $t'_1,\ldots,t'_{n-1}$ contain exactly one nonterminal symbol, and $t'_1,\ldots,t'_n$ do not contain any language name. □

We remark that the "extended lb grammars" of [5] and the "regularly extended lb grammars" in [10] have been defined in a way such that (implicitly) the OI variant has been chosen.

We provide K-elb grammars with a control mechanism in order to make some proofs more transparent, and to compare the generative capacity of controlled K-elb grammars with controlled K-iteration grammars; cf. Section 3. Since productions of the form 2.1(iii) may actually be thought to be substituted in productions of the forms 2.1(i) and 2.1(ii) (cf. [5, 10]) we label only rules of the form 2.1(i) and 2.1(ii), specify a language M over the set of these labels, and only allow derivations according to this

control language M.

We write $\sigma \overset{\pi}{\underset{m}{\Longrightarrow}} \tau$, where $\pi = \ell_1\ell_2\ldots\ell_k$, to denote an m-derivation which, disregarding rules of the form 2.1(iii), successively uses the rules labeled by $\ell_1, \ell_2, \ldots$ and ℓ_k (For the structure of such derivations, cf. Remarks 2.2 and 2.3).

Definition 2.4. Let Γ and K be families of languages. An m-extended Γ-controlled linear basic K-grammar or (m,Γ,K)-elb grammar is an 8-tuple $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ where

- Λ is a finite set of labels
- (Φ,Ψ,Σ,X,P,S) is an (m,K)-elb grammar in which each production of the forms 2.1(i) or 2.1(ii) is uniquely labeled with an element of Λ
- $M \subseteq \Lambda^*$ is a language in the family Γ (the control language of G).

The language generated by G is defined by $L_m(G) = \{w \in \Sigma^* \mid S \overset{\pi}{\underset{m}{\Longrightarrow}} w$ for some $\pi \in M\}$. □

The family of languages generated by (m,Γ,K)-elb grammars is denoted by $LB_m(\Gamma,K)$.

In order to avoid trivialities we always assume that the family Γ is nontrivial, i.e. Γ contains a nonempty language unequal to $\{\lambda\}$.

We will consider an alternative description of the generation process of a (controlled) elb grammar which corresponds closely to the discussion in the introduction. To this end we need the well-known concept of substitution which will often be called nondeterministic substitution in the sequel, in order to distinguish it from its deterministic variant (cf. [4]). The notion of deterministic substitution will frequently be used in Section 3.

Definition 2.5. Let K be a family. A (nondeterministic) K-substitution or nK-substitution τ on an alphabet Δ (without rank) is a mapping $\tau : \Delta \to K$. This mapping is defined for words by $\tau(\lambda) = \{\lambda\}$, $\tau(\alpha_1.....\alpha_n) = \tau(\alpha_1)...\tau(\alpha_n)$ for each $\alpha_1...\alpha_n \in \Delta^*$, and for languages by $\tau(L) = \cup \{\tau(w) \mid w \in L\}$ for each $L \subseteq \Delta^*$.

A deterministic K-substitution or dK-substitution τ on Δ is also a mapping $\tau : \Delta \to K$. It is defined for words $w \in \Delta^*$ by $\tau(w) = \{h(w) \mid h$ is a homomorphism such that $h(\alpha) \in \tau(\alpha)$ for each $\alpha \in \Delta\}$, and for languages by $\tau(L) = \cup \{\tau(w) \mid w \in L\}$ for each L over Δ. □

Thus deterministically substituting languages $\tau(\alpha)$ from K in a word $w \in \Delta^*$ means replacing α by the same word $u \in \tau(\alpha)$ for all occurrences of α in w. (It may be assumed that $\tau(\alpha) \neq \emptyset$ for each α in Δ [4]).

We are now ready for the alternative derivation relations.

Definition 2.6. Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be an (m,Γ,K)-elb grammar.

(1) An OI-configuration is either a tuple of the form $\langle A,L_1,...,L_n\rangle$ with $A \in \Phi_n$ and $L_i \subseteq \Sigma^*$ $(n \geq 0,\ 1 \leq i \leq n)$, or a language $L \subseteq \Sigma^*$. The OI-derivation relation $\vdash_{OI}$ between OI-configurations is defined as follows.

- If $\ell : A(x_1,...,x_n) \to B(\psi_1(\underline{x}),...,\psi_k(\underline{x}))$ and $\psi_i(\underline{x}) \to L_i'$ $(1 \leq i \leq k)$ are in P, then $\langle A,L_1,...,L_n\rangle \vdash_{OI}^{\ell} \langle B,\tau(L_1'),...,\tau(L_k')\rangle$ where the nK-substitution τ on $\Sigma \cup \{x_1,...,x_n\}$ is defined by $\tau(x_i) = L_i$ and $\tau(a) = \{a\}$ for each a in Σ.
- If $\ell : A(x_1,...,x_n) \to \psi(x_1,...,x_n)$ and $\psi(x_1,...,x_n) \to L'$ are in P, then $\langle A,L_1,...,L_n\rangle \vdash_{OI}^{\ell} \tau(L')$ where the nK-substitution τ is defined as above.

(2) An IO-configuration is either a tuple $\langle A,u_1,...,u_n\rangle$ with $A \in \Phi_n$ and $u_i \in \Sigma^*$ $(n \geq 0,\ 1 \leq i \leq n)$, or a word $u \in \Sigma^*$. The IO-derivation relation $\vdash_{IO}$ between IO-configurations is defined as follows.

- If $\ell : A(x_1,\ldots,x_n) \to B(\psi_1(\underline{x}),\ldots,\psi_k(\underline{x}))$ and $\psi_i(\underline{x}) \to L_i'$ $(1 \le i \le k)$ are in P then $\langle A,u_1,\ldots,u_n\rangle \underset{IO}{\overset{\ell}{\vdash}} \langle B,v_1,\ldots,v_k\rangle$ for each $v_i \in h(L_i')$, $1 \le i \le k$, where h is the homomorphism on $\Sigma \cup \{x_1,\ldots,x_n\}$ defined by $h(x_i) = u_i$ and $h(a) = a$ for each a in Σ.
- If $\ell : A(x_1,\ldots,x_n) \to \psi(x_1,\ldots,x_n)$ and $\psi(x_1,\ldots,x_n) \to L'$ are in P, then $\langle A,u_1,\ldots,u_n\rangle \underset{IO}{\overset{\ell}{\vdash}} v$ for each $v \in h(L')$ where the homomorphism h is defined as above.

The relations $\underset{OI}{\overset{\pi}{\vdash}}$ and $\underset{IO}{\overset{\pi}{\vdash}}$ are defined in the obvious way. □

The following lemma expresses the equivalence of the derivation relations $\underset{m}{\vdash}$ and $\underset{m}{\Longrightarrow}$.

Lemma 2.7. Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be an (m,Γ,K)-elb grammar. Then

(1) $L_{OI}(G) = \cup \{L \subseteq \Sigma^* \mid \langle S\rangle \underset{OI}{\overset{\pi}{\vdash}} L \text{ for some } \pi \in M\}$

(2) $L_{IO}(G) = \{w \in \Sigma^* \mid \langle S\rangle \underset{IO}{\overset{\pi}{\vdash}} w \text{ for some } \pi \in M\}$.

Proof. (1) By a straightforward induction argument it can be shown that $\langle S\rangle \underset{OI}{\overset{\pi}{\vdash}} \langle A,L_1,\ldots,L_n\rangle$ iff there exist (unique) terms $t_1,\ldots,t_n$ over $\Psi \cup \Sigma$ such that $S \underset{OI}{\overset{\pi}{\Longrightarrow}} A(t_1,\ldots,t_n)$ and $L_i = \{w \in \Sigma^* \mid t_i \underset{OI}{\overset{*}{\Longrightarrow}} w\}$. From this it easily follows that, for $w \in \Sigma^*$, $S \underset{OI}{\overset{\pi}{\Longrightarrow}} w$ iff there exists an L such that $\langle S\rangle \underset{OI}{\overset{\pi}{\vdash}} L$ and $w \in L$.

(2) In this case it can be shown that $\langle S\rangle \underset{IO}{\overset{\pi}{\vdash}} \langle A,w_1,\ldots,w_n\rangle$ iff $S \underset{IO}{\overset{\pi}{\Longrightarrow}} A(w_1,\ldots,w_n)$ and $\langle S\rangle \underset{IO}{\overset{\pi}{\vdash}} w$ iff $S \underset{IO}{\overset{\pi}{\Longrightarrow}} w$. □

We mention here that considering τ to be a dK-substitution in Definition 2.6(1) corresponds (in the sense of Lemma 2.7) to derivations of the form $S \underset{OI}{\overset{*}{\Longrightarrow}} t_f \underset{IO}{\overset{*}{\Longrightarrow}} w$ (cf. Remark 2.2), i.e. derivations in which first rules of the form 2.1(i) and 2.1(ii) are applied (in the OI-way) and then, as soon as the nonterminal disappears, productions of the form 2.1(iii) are used in the IO-way.

In order to facilitate establishing equivalence theorems and to show the close relationship to register programs we introduce (m,Γ,K)-elb grammars of a special type, viz. in normal form (Definition 2.8). Then we show that under rather weak assumptions on the families Γ and K, each (m,Γ,K)-elb grammar can be transformed into a grammar in normal form (Lemma 2.9) generating the same language.

Before turning to normal forms we introduce some additional terminology. For each elb grammar, a production is initial if its left-hand side is equal to the initial nonterminal S. Productions of the form 2.1(i) with left-hand side unequal to S are called intermediate productions, whereas a production of the form 2.1(ii) is terminal.

Definition 2.8. An (m,Γ,K)-elb grammar $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ is in normal form if there exists a natural number $n \geq 1$ such that

(1) $\Phi = \Phi_0 \cup \Phi_n$ with $\Phi_0 = \{S\}$ and Φ_n is a singleton (So $\Phi_p = \emptyset$ for p unequal to 0 or n).

(2) $X = \{x_1,\ldots,x_n\}$.

(3) $\Psi = \Psi_0 \cup \Psi_n$, i.e. $\Psi_p = \emptyset$ for p unequal to 0 or n.

(4) The productions of type 2.1(i) and 2.1(ii) have the following forms

$S \to E(\psi_{01},\ldots,\psi_{0n})$ (initial production)

$E(\underline{x}) \to E(\psi_1(\underline{x}),\ldots,\psi_n(\underline{x}))$ (intermediate production)

$E(\underline{x}) \to \psi(\underline{x})$ (terminal production)

where $\Phi_n = \{E\}$, $(\underline{x}) = (x_1,\ldots,x_n)$; $\psi_{01},\ldots,\psi_{0n} \in \Psi_0$, and $\psi,\psi_1,\ldots,\psi_n \in \Psi_n$. □

Lemma 2.9. (Normal Form Lemma). Let K be a family including the family SYMBOL and let Γ be a family closed under intersection with regular sets. Then for each (m,Γ,K)-elb grammar G there exists an (m,Γ,K)-elb grammar H in normal form such that $L_m(H) = L_m(G)$.

Proof. Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be an (m,Γ,K)-elb grammar with $X = \{x_1,\ldots,x_n\}$. Assume that $\Sigma \neq \emptyset$ and let σ be an arbitrary but fixed element in Σ.

First we change the rank of all language names, apart from those occurring in initial productions, and of all nonterminals in $\Phi - \{S\}$ to n as follows:

(1) By adding (dummy) arguments to the argument list the rank of each language name - with the exception of those occurring in initial productions - is enlarged from k ($0 \leq k < n$) to n. E.g. $\psi(x_1,\ldots,x_k) \to L$ is replaced by $\psi(x_1,\ldots,x_n) \to L$. Note that L is a language over $\Sigma \cup \{x_1,\ldots,x_k\}$.

(2) In each noninitial production we extend the rank of the nonterminals to n (whenever this is necessary). The new open places in the argument list are filled up with (dummy) arguments in the left-hand side and with $\psi_e(x_1,\ldots,x_n)$ in the right-hand side. Here ψ_e is a new language name of rank n, for which we add the rule $\psi_e(x_1,\ldots,x_n) \to \{\sigma\}$ to P (Note that $\{\sigma\}$ is in K). Thus a production like e.g.

$A(x_1,\ldots,x_k) \to B(\psi_1(\underline{x}),\ldots,\psi_p(\underline{x}))$ (*)

with $(\underline{x}) = (x_1,\ldots,x_k)$ and $k,p \leq n$, is replaced by

$A(x_1,\ldots,x_n) \to B(\psi_1(\underline{x}),\ldots,\psi_p(\underline{x}),\psi_e(\underline{x}),\ldots,\psi_e(\underline{x}))$ (**)

where $(\underline{x}) = (x_1,\ldots,x_n)$ and both A and B now have rank n.

(3) We replace each initial rule of the form $S \to A(\psi_1,\ldots,\psi_k)$, A has rank k ($0 \leq k \leq n$) and $\psi_1,\ldots,\psi_k \in \Psi_0$, by a corresponding initial production $S \to A(\psi_1,\ldots,\psi_k,\psi_0,\ldots,\psi_0)$ where A has now rank n. Here ψ_0 is a new language name of rank 0 with the (new) production $\psi_0 \to \{\sigma\}$ in P.

Note that, if we apply (**) instead of (*) we actually have to know only the values of the first k arguments of A, because $\psi_1,\ldots,\psi_p$ still represent languages over $\Sigma \cup \{x_1,\ldots,x_k\}$, and the language represented by ψ_e is over Σ, i.e. no variable is involved at all. Also the introduction of

the language $\{\sigma\}$ does not produce any new string in the language generated by the modified grammar (In the OI case any other language $L \subseteq \Sigma^*$ in K could be used instead of $\{\sigma\}$; in the IO case the same is true for nonempty L). By these properties the (m,Γ,K)-elb grammar obtained by the modifications (1) - (3) still generates the same language.

The grammar obtained so far is almost in normal form; we only have to change each symbol from $\Phi - \{S\}$ into a single nonterminal E of rank n. The loss of finite state control (i.e. a regular language R over Λ) due to this replacement can however be completely disposed of by a new control language $M \cap R$. More precisely, let R be the regular language accepted by the deterministic finite state acceptor $(Q,\Lambda,\delta,S,\{q_f\})$ where $Q = \Phi \dot{\cup} \{q_f\}$ (In this context Φ is assumed to be a non-ranked set). is the set of states, Λ is the alphabet, S is the initial state, $\{q_f\}$ is the set of final states, and the transition function $\delta : Q \times \Lambda \to Q$ is defined by

$\delta(A,\ell) = B$ if $\ell : A(\ldots) \to B(\ldots)$ is in P,

$\delta(A,\ell) = q_f$ if the production labeled by ℓ is terminal.

Let the (m,Γ,K)-elb grammar H be defined by $H = (\Phi',\Psi',\Sigma,X,P',S,\Lambda,M')$ where Φ',Ψ' and P' are obtained from Φ, Ψ and P as described above, and $M' = M \cap R$. Then H is in normal form and $L_m(H) = L_m(G)$. □

Just as (in the previous lemma) the use of nonterminals can be replaced by regular control, we can show (in the next lemma) that regular control can be simulated by "storing it in the nonterminals".

Lemma 2.10. (Regular Control Lemma). $LB_m(REG,K) = LB_m(K)$.

Proof. Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be an (m,REG,K)-elb grammar. Assume that the control language $M \subseteq \Lambda^*$ is accepted by the deterministic finite state

acceptor $(Q,\Lambda,\delta,q_0,Q_f)$ where Q, $\delta : Q \times \Lambda \to Q$, $q_0 \in Q$ and $Q_f \subseteq Q$ are respectively the set of states, the transition function, the initial state, and the set of final states.

Consider the uncontrolled (m,K)-elb grammar $H = (\Phi',\Psi,\Sigma,X,P',S')$ where $\Phi' = \Phi \times Q$ and $[A,q]$ has the same rank as A, $S' = [S,q_0]$ and P' is obtained from P as follows:

- If $\ell : A(\ldots) \to B(\ldots)$ is a rule in P and $\delta(q_1,\ell) = q_2$, then the set P' contains the production $[A,q_1](\ldots) \to [B,q_2](\ldots)$.
- If $\ell : A(\ldots) \to \psi(\ldots)$ is a terminal rule in P and $\delta(q,\ell) \in Q_f$, then the set P' contains the production $[A,q](\ldots) \to \psi(\ldots)$.

From this construction it easily follows that $L_m(H) = L_m(G)$. Hence $LB_m(REG,K) \subseteq LB_m(K)$. The converse inclusion is obvious. □

We conclude this section with a concrete example.

Let G be the (m,REG,FIN)-elb grammar in normal form with the following productions:

$\ell_0 : S \to A(\psi_{01},\psi_{02})$

$\ell_1 : A(x,y) \to A(\psi_{11}(x,y),\psi_{12}(x,y))$

$\ell_2 : A(x,y) \to \psi_2(x,y)$

$\psi_{01} \to \{a,b\}$; $\psi_{11}(x,y) \to \{xyx\}$; $\psi_2(x,y) \to \{xy\}$

$\psi_{02} \to \{c,d\}$; $\psi_{12}(x,y) \to \{y\}$;

and control language $M = \{\ell_0\ell_1^{2k+1}\ell_2 \mid k \geq 0\}$.

One can easily show that

(1) $L_{OI}(G) = \cup \{(\{a,b\}\{c,d\})^n \mid n = 2^{2k+1};\ k \geq 0\}$

(2) $L_{IO}(G) = \{w^n \mid n = 2^{2k+1};\ k \geq 0;\ w \in \{ac,ad,bc,bd\}\}$

(3) $L_{IO}(G) \subset L_{OI}(G)$, whereas the language L obtained by the derivation relation of Definition 2.6(1) in which τ is interpreted as a dK-substitution (instead of an nK-substitution) properly lies between $L_{IO}(G)$ and $L_{OI}(G)$.

In fact acad $\in$ L, but no word in L contains both a and b.

(4) The uncontrolled (m,FIN)-elb grammar with rules

$S \to A_1(\psi_{01},\psi_{02})$

$A_1(x,y) \to A_2(\psi_{11}(x,y),\psi_{12}(x,y))$

$A_2(x,y) \to A_1(\psi_{11}(x,y),\psi_{12}(x,y))$

$A_2(x,y) \to \psi_2(x,y)$,

where the productions for the language names are as above, also generates $L_m(G)$ (cf. Lemma 2.10).

3. Relation to Iteration Grammars

Iteration grammars have been introduced in [27, 26] to provide a unifying framework for investigating context-independent L systems and their connection to iterated substitution.

First, we quote a few concepts from [27, 26, 3, 4] collected in the following definition.

Definition 3.1. Let Γ and K be families of languages, and let m be either n or d (where n and d abbreviate nondeterministic and deterministic respectively).

An (m,K)-iteration grammar G is a 5-tuple (V,Σ,U,I,S) where

- V is an alphabet
- $\Sigma \subseteq V$ is the terminal alphabet of G
- $S \in V$ is the initial symbol
- I is a finite set, called the index alphabet of G
- $U = \{\tau_i \mid i \in I\}$ is a finite set of mK-substitutions on V such that $\tau_i(\alpha)$ is a language over V for each $i \in I$ and each $\alpha \in V$.

For $\pi = i_1 i_2 \ldots i_p$ and $L \subseteq V^*$ we write $\tau_\pi(L)$ as an abbreviation of $\tau_{i_p} \ldots \tau_{i_2} \tau_{i_1}(L)$.

The language $L_m(G)$ generated by G is defined by $L_m(G) = \cup \{\tau_\pi(S) \mid \pi \in I^*\} \cap \Sigma^*$.

An (m,Γ,K)-iteration grammar G is a 6-tuple (V,Σ,U,I,M,S) where

- (V,Σ,U,I,S) is an (m,K)-iteration grammar.
- $M \subseteq I^*$ is a language in Γ, called the control language of G.

The language $L_m(G)$ generated by G is defined by $L_m(G) = \Sigma^* \cap \cup \{\tau_\pi(S) \mid \pi \in M\}$.

The family of languages generated by (n,Γ,K)-iteration [(n,K)-iteration, (d,Γ,K)-iteration, (d,K)-iteration] grammars is denoted by H(Γ,K) [H(K), η(Γ,K), η(K)]. □

In propositions and proofs the prefix "m" is used as follows: either m refers to OI (in the context of elb grammars) and n (as prefix for iteration grammars) or to IO and d respectively.

In this section we establish one of our main results, viz. we prove that (under certain assumptions on the families Γ and K) the language family generated by (m,Γ,K)-elb grammars precisely coincides with the family of (m,Γ,K)-iteration languages (Theorem 3.4). Since for elb grammars (Lemma 2.10) as well as for iteration grammars [3,4] regular control does not increase the generating capacity of the uncontrolled devices, we obtain a similar equivalence result for the uncontrolled case (Theorem 3.5).

Our proof of this equivalence theorem is divided into the following two lemmas for which we need some additional notation and terminology.

Remember that a family Γ is closed under full marking if for each language $M \subseteq \Lambda^*$ in Γ and for all symbols $\alpha, \beta \notin \Lambda$, the languages αM and Mβ are also in Γ.

Let ρ be the reversal operation, i.e. the operation satisfying $\rho(\lambda) = \lambda$, $\rho(\alpha_1\alpha_2\ldots\alpha_{n-1}\alpha_n) = \alpha_n\alpha_{n-1}\ldots\alpha_2\alpha_1$, and $\rho(L) = \{\rho(w) \mid w \in L\}$.

Lemma 3.2. Let K be a family including SYMBOL, and let Γ be a family closed under reversal and intersection with regular sets. Then

$LB_{IO}(\Gamma,K) \subseteq \eta(\Gamma,K)$, and $LB_{OI}(\Gamma,K) \subseteq H(\Gamma,K)$.

Proof. Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be an (m,Γ,K)-elb grammar. By 2.9 we assume that G is in normal form. So let $\Phi = \{S,E\}$, where E has rank n, and let $X = \{x_1,\ldots,x_n\}$.

Furthermore, as Γ is closed under intersection with regular sets we may also assume that each control word in M starts with a label of an initial production and it ends with a label of a terminal rule, whereas only labels of intermediate productions occur between the first and the last symbol of a control word.

We define an (m,Γ,K)-iteration grammar H which contains for each production of G involving a nonterminal symbol (i.e. for each rule of the form 2.1(i) or 2.1(ii)) a corresponding mK-substitution. The construction is such that H simulates each sequence of productions from G in reversed order so that an initial production of G corresponds to a terminal mK-substitution in H, and vice versa.

Let $H = (V,\Sigma,U,I,M_H,S_H)$ where $V = \Sigma \cup X$, $S_H = x_1$, $I = \Lambda$ and $M_H = \rho(M)$. The set U of mK-substitutions is obtained from the productions in P as follows:

(1) If $\ell : E(\underline{x}) \to \psi(\underline{x})$ is a terminal production in P with the (unique) rule $\psi(\underline{x}) \to L$ for ψ $(L \subseteq V^*)$, then we define an mK-substitution $\tau_\ell \in U$ by

$\tau_\ell(x_1) = L$

$\tau_\ell(\alpha) = \{\alpha\}$ for each α in $V - \{x_1\}$

(2) For each intermediate production in P of the form

$\ell : E(\underline{x}) \to E(\psi_1(\underline{x}),\ldots,\psi_n(\underline{x}))$ with for each ψ_i $(1 \le i \le n)$ the (unique) rule $\psi_i(\underline{x}) \to L_i$ in P $(L_i \subseteq V^*)$, U contains an mK-substitution τ_ℓ defined by

$\tau_\ell(x_i) = L_i$ for each i, $1 \le i \le n$

$\tau_\ell(\alpha) = \{\alpha\}$ for each α in Σ

(3) With each initial production in P, $\ell : S \to E(\psi_1,\ldots,\psi_n)$ where $\psi_i \to L_i$ ($1 \le i \le n$; $L_i \subseteq \Sigma^*$) is the unique rule for ψ_i in P, we associate an mK-substitution $\tau_\ell \in U$ defined by

$\tau_\ell(x_i) = L_i$ for each i, $1 \le i \le n$

$\tau_\ell(\alpha) = \{\alpha\}$ for each α in Σ

We now prove the correctness of this construction. First, we consider the OI/nondeterministic case; then we discuss the IO/deterministic variant.

In the OI-case, the elb-grammar G keeps in the i-th argument of E the set of all words which can be derived (according to some control word $\rho(\pi)$) from x_i in H. To compute this set (for each i) G uses the control word π. Formally we show by induction that, for each proper prefix π of a control word in M (cf. the assumption on M), $\langle S\rangle \vdash_{OI}^{\pi} \langle E,\tau_{\rho(\pi)}(x_1),\ldots,\tau_{\rho(\pi)}(x_n)\rangle$.

Initial step: Let $\pi = \ell$ and let ℓ be the label of the initial production $\ell : S \to E(\psi_1,\ldots,\psi_n)$ where $\psi_i \to L_i$ ($L_i \subseteq \Sigma^*$, $1 \le i \le n$) is the production for ψ_i in P. From Definition 2.6(1) and from (3) above it directly follows that $\langle S\rangle \vdash_{OI}^{\ell} \langle E,L_1,\ldots,L_n\rangle = \langle E, \tau_\ell(x_1),\ldots,\tau_\ell(x_n)\rangle$.

Induction step: Suppose $\langle S\rangle \vdash_{OI}^{\pi} \langle E,\tau_{\rho(\pi)}(x_1),\ldots,\tau_{\rho(\pi)}(x_n)\rangle$. Let ℓ be the label of the intermediate production $\ell : E(x_1,\ldots,x_n) \to E(\psi_1(\underline{x}),\ldots,\psi_n(\underline{x}))$, and let $\psi_i(\underline{x}) \to L_i$ with $L_i \subseteq (\Sigma \cup X)^*$ be the production for ψ_i in P ($1 \le i \le n$). From the induction hypothesis, Definition 2.6(1) and from (2) above we obtain $\langle S\rangle \vdash_{OI}^{\pi} \langle E,\tau_{\rho(\pi)}(x_1),\ldots,\tau_{\rho(\pi)}(x_n)\rangle \vdash_{OI}^{\ell} \langle E, \tau(L_1),\ldots,\tau(L_n)\rangle$ where $\tau = \tau_{\rho(\pi)}$. This last configuration is equal to $\langle E,\tau\tau_\ell(x_1),\ldots,\tau\tau_\ell(x_n)\rangle = \langle E,\tau_{\rho(\pi\ell)}(x_1),\ldots,\tau_{\rho(\pi\ell)}(x_n)\rangle$.

Finally, suppose $\ell : E(x_1,\ldots,x_n) \to \psi(\underline{x})$ is a terminal rule and $\pi\ell \in M$. If $\psi(\underline{x}) \to L$ is the rule for ψ in P, then it is clear from Definition 2.6(1) and from (1) above that $\langle S\rangle \vdash_{OI}^{\pi} \langle E,\tau_{\rho(\pi)}(x_1),\ldots,\tau_{\rho(\pi)}(x_n)\rangle \vdash_{OI}^{\ell} \tau_{\rho(\pi)}(L) = \tau_{\rho(\pi)}\tau_\ell(x_1) = \tau_{\rho(\pi\ell)}(x_1)$. Hence, for $\pi \in M$, $\langle S\rangle \vdash_{OI}^{\pi} \tau_{\rho(\pi)}(x_1)$; note that, by (3), $\tau_{\rho(\pi)}(x_1) \subseteq \Sigma^*$. Since for a given π there is only one derivation $\langle S\rangle \vdash_{OI}^{\pi} L$,

it follows that $L_{OI}(G) = \cup \{\tau_{\rho(\pi)}(x_1) \mid \pi \in M\} = L_n(H)$.

This completes the proof of the OI/nondeterministic case.

In the IO-case, the elb grammar G keeps in the arguments of E a sequence of words $u_1,\ldots,u_n$ which can be derived from $x_1,\ldots,x_n$ (respectively) by some control word $\rho(\pi)$, in such a way that, at each moment of time, each occurrence of the same symbol (in all the derivations) is replaced by the same word. This consistency condition can be expressed by saying that $u_1u_2\ldots u_n$ can be derived from $x_1x_2\ldots x_n$. We leave it to the reader to prove formally that $\langle S\rangle \overset{\pi}{\underset{IO}{\vdash}} \langle E,u_1,\ldots,u_n\rangle$ iff (a) $u_i \in \tau_{\rho(\pi)}(x_i)$ for $1 \leq i \leq n$, and (b) $u_1\ldots u_n \in \tau_{\rho(\pi)}(x_1\ldots x_n)$. In that proof the consistency requirement can be dealt with using Lemmas 5.1 and 5.2 of [4]. From this it easily follows that, for all $\pi \in M$, $\langle S\rangle \overset{\pi}{\underset{IO}{\vdash}} u$ iff $u \in \tau_{\rho(\pi)}(x_1)$; note that, by (3), $u \in \Sigma^*$. Hence $L_{IO}(G) = \cup \{\tau_{\rho(\pi)}(x_1) \mid \pi \in M\} = L_d(H)$.

This completes the proof of the IO/deterministic case. □

An isomorphism (or "renaming of symbols") is a one-to-one SYMBOL-substitution.

Lemma 3.3. Let K be a family including SYMBOL $\cup \{\emptyset\}$ and closed under isomorphism, and let Γ be a family closed under full marking and reversal. Then

(1) $H(\Gamma,K) \subseteq LB_{OI}(\Gamma,K)$,

(2) if Γ is closed under finite substitution, and if both Γ and K are closed under intersection with regular sets, then $\eta(\Gamma,K) \subseteq LB_{IO}(\Gamma,K)$.

Proof. Let $G = (V,\Sigma,U,I,M,S)$ be an (m,Γ,K)-iteration grammar, where $U = \{\tau_i \mid i \in I\}$ and $V = \{\alpha_1,\ldots,\alpha_n\}$ with $S = \alpha_1$.
We first prove the special case that $\Sigma = V$. Then we consider the effect

of a (possibly proper) terminal alphabet.

We construct an (m,Γ,K)-elb grammar H in normal form which simulates each sequence of substitutions from G in reversed order. Let H be defined by $H = (\Phi,\Psi,V,X,P,S_H,\Lambda,M_H)$ where $\Phi = \{S,E\}$ (E has rank n); $\Psi = \Psi_0 \cup \Psi_n$ with $\Psi_0 = \{\psi_j \mid 1 \leq j \leq n\}$ and $\Psi_n = \{\psi\} \cup \{\psi_{ij} \mid i \in I;\ 1 \leq j \leq n\}$. Furthermore, let $X = \{x_1,\ldots,x_n\}$, $\Lambda = I \cup \{s,t\}$ where s and t are new symbols, and $M_H = s\rho(M)t$. The set P consists of the following productions:

s: $S \to E(\psi_1,\ldots,\psi_n)$

$\psi_j \to \{\alpha_j\}$ for each j, $1 \leq j \leq n$

i: $E(\underline{x}) \to E(\psi_{i1}(\underline{x}),\ldots,\psi_{in}(\underline{x}))$ for each $i \in I$

$\psi_{ij}(\underline{x}) \to L_{ij}$ for each $i \in I$ and each j, $1 \leq j \leq n$;

where L_{ij} is the result of renaming each α_p by x_p in $\tau_i(\alpha_j)$, $1 \leq p \leq n$

t: $E(\underline{x}) \to \psi(\underline{x})$

$\psi(\underline{x}) \to \{x_1\}$

Applying to H the construction exhibited in the proof of Lemma 3.2 again yields an (m,Γ,K)-iteration grammar for which one can easily see that it generates $L_m(G)$. This fact implies the correctness of the construction of H in the present proof.

We now turn to the general case, viz. $\Sigma \subseteq V$ with $\Sigma = \{\alpha_{k+1},\ldots,\alpha_n\}$ for some k $(1 \leq k \leq n)$.

In the nondeterministic/OI case the language $L_{OI}(H) \cap \Sigma^*$ can easily be obtained by substituting $\emptyset$ for each element of $V - \Sigma$: change the production in P labeled by s together with the corresponding rules for ψ_j into

s: $S \to E(\psi_1,\ldots,\psi_n)$

$\psi_j \to \emptyset$ for each j, $1 \leq j \leq k$

$\psi_j \to \{\alpha_j\}$ for each j, $k+1 \leq j \leq n$.

Since an IO derivation is blocked as soon as the empty language occurs as value in an argument list, we cannot apply this method in the deterministic/IO case. Instead we show that for each (d,Γ,K)-iteration grammar $G = (V,\Sigma,U,I,M,S)$

there exists an equivalent one with $\Sigma = V$. Let alph(w) be the set of all symbols occurring in w. Since Γ is closed under finite substitution and K under intersection with regular sets, we may assume according to Lemma 3.1 of [4] that G has the following property: for each $\tau_i \in U$ and each $\alpha \in V$, if w_1 and w_2 are in $\tau_i(\alpha)$, then $alph(w_1) = alph(w_2)$. It is easy to show that the same property holds for arbitrary τ_π, $\pi \in I^*$. In fact, if we denote by alph(L) the unique alph(w) for $w \in L$, then $alph(\tau_i\tau_\pi(\alpha)) = \cup \{alph(\tau_i(\beta)) \mid \beta \in alph(\tau_\pi(\alpha))\}$. Hence according to ah arbitrary control word π either only terminal strings are produced, or only strings which are not terminal are obtained. Consequently, by intersecting M with the set $D = \{\pi \in I^* \mid alph(\tau_\pi(\alpha_1)) \subseteq \Sigma\}$, only terminal strings are produced, and the terminal alphabet is made superfluous. Thus G is equivalent with $G' = (V,V,U,I,M \cap D,S)$. It should be clear from the above formula that D is regular. □

From these lemmas we now obtain the main result of this section.

Theorem 3.4. (Equivalence Theorem). Let K be a family including SYMBOL $\cup \{\emptyset\}$ and closed under isomorphism, and let Γ be a family closed under reversal and under intersection with regular sets.

(1) If Γ is closed under full marking, then $LB_{OI}(\Gamma,K) = H(\Gamma,K)$.

(2) If Γ is closed under finite substitution, and if K is closed under intersection with regular sets, then $LB_{IO}(\Gamma,K) = \eta(\Gamma,K)$.

Proof. Lemmas 3.2, 3.3 and the fact that each family closed under finite substitution and intersection with regular sets is also closed under full marking. □

As a particular case we have a similar result for the corresponding uncontrolled families.

Theorem 3.5. Let K be a family including ONE ∪ {∅} and closed under isomorphism. Then

(1) $LB_{OI}(K) = H(K)$;

(2) if K is closed under intersection with regular sets, then $LB_{IO}(K) = \eta(K)$.

Proof. The statements directly follow from Theorem 2.1 in [3], Theorem 2.7 in [4], Lemma 2.10 and Theorem 3.4. □

Theorem 3.5 is the essential generalization of two main results from [5] which may be presented in our notation as $LB_{IO}(ONE \cup \{\emptyset\}) = EDTOL$ $(= \eta(ONE \cup \{\emptyset\}))$ and $LB_{OI}(FIN) = ETOL$ $(= H(FIN))$. In addition we note that from Lemma 2.4 in [4] it follows that $LB_{IO}(FIN) = \eta(FIN) = EDTOL = LB_{IO}(ONE \cup \{\emptyset\})$.

Both Theorems 3.4 and 3.5 enable us to establish properties for LB_{OI} and LB_{IO} by means of results already obtained for H and η respectively. As an example we consider closure properties of $LB_m(\Gamma,K)$ and $LB_m(K)$ for which we recall some terminology.

A full Quasi Abstract Family of Languages or full QAFL is a family including SYMBOL and closed under homomorphism, intersection with regular sets and the regular operations (i.e. union, concatenation and Kleene star). So a full AFL is a full QAFL closed under inverse homomorphism. A full dhyper-QAFL [full hyper-AFL] is a full QAFL [full AFL] closed under iterated [non]deterministic substitution, i.e. closed under η [H]; cf. [27, 26, 3, 4].

Combining Theorem 3.4 and 3.5 with Theorems 4.4 and 6.3 in [3] and Theorems 4.3 and 4.4 in [4] yields the following corollaries.

Corollary 3.6. Let Γ and K satisfy the conditions of the Equivalence Theorem 3.4(1). Let in addition K be closed under finite substitution and intersection with regular sets. Then

(1) $LB_{OI}(K)$ is the smallest full hyper-AFL including K;

(2) if Γ is closed under union (or concatenation) and Kleene star, then $LB_{OI}(\Gamma,K)$ is a full hyper-AFL including Γ, $LB_{OI}(K)$ and ETOL. □

Corollary 3.7. Let Γ and K satisfy the conditions of the Equivalence Theorem 3.4(2). Let in addition K be closed under homomorphism. Then

(1) $LB_{IO}(K)$ is the smallest full dhyper-QAFL including K;

(2) if Γ is closed under union (or concatenation) and Kleene star, then $LB_{IO}(\Gamma,K)$ is a full dhyper-QAFL including Γ, $LB_{IO}(K)$ and EDTOL. □

According to Theorem 3.5 or to Corollaries 3.6(1) and 3.7(1) a full QAFL [full AFL] K is a full dhyper-QAFL [full hyper-AFL] iff $LB_{IO}(K) \subseteq K$ [$LB_{OI}(K) \subseteq K$]. This characterization is particularly useful in proving (sub)classes of macro languages to be a full dhyper-QAFL or a full hyper-AFL. As an example we show (the known facts) that the family OI is a full hyper-AFL [27, 26, 5] and that the family IO is a full dhyper-QAFL [4]. The facts that the family IO is a full QAFL and the family OI is a full AFL have been proved in [12]. So it remains to show that $LB_{IO}(IO) \subseteq IO$ and $LB_{OI}(OI) \subseteq OI$.

Let $G = (\Phi,\Psi,\Sigma,X,P,S)$ be an (m,K_m)-elb grammar, where K_m is the family IO if m = IO, and similarly for m = OI. Consider rules in G of the form $A(x_1,\ldots,x_n) \to B(\psi_1(\underline{x}),\ldots,\psi_k(\underline{x}))$ and $A(x_1,\ldots,x_n) \to \psi_0(\underline{x})$ with $\psi_i(\underline{x}) \to L_i$ $(0 \le i \le k)$. Let L_i be generated by an m-macro grammar G_i (Note that $x_1,\ldots,x_n$ are terminal symbols of G_i). We may assume that both the nonterminals and the variables of G and all the G_i's are mutually distinct.

We construct a macro grammar H from G and all the G_i's as follows. The productions in G of the form 2.1(i) and 2.1(ii) are also rules in H (The language names of G are treated in H as ordinary nonterminals). For each production of G_i we add a corresponding rule to H which is obtained by

extending the argument list of each (possibly nested) occurrence of a nonterminal with $x_1,\ldots,x_n$ (As an example, if $C(y_1,y_2) \rightarrow x_1^2C(x_1y_1,D(y_2a))by_2$ is in a G_i and if n = 2, then $C(y_1,y_2,x_1,x_2) \rightarrow x_1^2C(x_1y_1,D(y_2a,x_1,x_2),x_1,x_2)by_2$ is in H). Finally, for each rule of the form 2.1(iii) $\psi_i(x_1,\ldots,x_n) \rightarrow L_i$ in G there is in H a rule $\psi_i(x_1,\ldots,x_n) \rightarrow S_i(x_1,\ldots,x_n)$, where S_i is the initial nonterminal of G_i.

It should be clear that for the resulting grammar we have $L_m(H) = L_m(G)$. Hence $LB_m(K_m) \subseteq K_m$.

Finally, we return to the example at the end of Section 2. Applying the construction of Lemma 3.2 to G yields the (m,REG,FIN)-iteration grammar $H = (V,\Sigma,U,I,M_H,S_H)$ where $V = \Sigma \cup \{x,y\}$, $S_H = x$, $I = \{0,1,2\}$ and $M_H = \{21^{2k+1}0 \mid k \geq 0\}$. The elements in U are defined as follows (We only mention the instances different from the identity mapping):

$\tau_2(x) = \{xy\}$

$\tau_1(x) = \{xyx\}$

$\tau_0(x) = \{a,b\}$; $\tau_0(y) = \{c,d\}$

It is left to the reader to verify that the language L generated by G with the derivation relation of Definition 2.6(1) in which τ is interpreted as a dK-substitution (instead of an nK-substitution) is also generated by H, when we require the following additional condition on the derivations (obtained from the application of nondeterministic substitutions): for all symbols α in V, all occurrences of α descending from the same father node have to be rewritten according to the same subderivation.

4. Register programs

In this section we investigate the relationship between (extended, controlled) linear basic grammars and various types of register programs. We start with an informal discussion.

The simplest type of register program that we consider is a nondeterministic flowchart (without tests) with registers $x_1,\ldots,x_n$ ($n \geq 1$) containing strings over some alphabet Σ, and with three kinds of (deterministic) instructions:

(i) a start instruction of the form $\underline{\text{start}}(x_1:=v_1,\ldots,x_n:=v_n)$, where each v_i is a string over Σ,

(ii) a (parallel) assignment of the form $(x_1:=w_1,\ldots,x_n:=w_n)$, where each w_i is a string over $\Sigma \cup \{x_1,\ldots,x_n\}$, and

(iii) a halt instruction of the form $\underline{\text{halt}}(w)$, where w is a string over $\Sigma \cup \{x_1,\ldots,x_n\}$.

A computation of such a register program starts with a start instruction (initializing its registers with certain strings over Σ) and ends with a halt instruction $\underline{\text{halt}}(w)$, producing the string determined by w. The parallel assignments are executed in the obvious way. Thus the register program computes (generates) a language over Σ. As an example consider the register program of Fig.1 with registers x,y,z and $\Sigma = \{a,b,c\}$. It should be clear that this program generates the language $\{(a^n b^n c^n)^m \mid n,m \geq 0\}$. Note that when in a parallel assignment the i-th item has been omitted, it is meant to be the identity $x_i:=x_i$.

From a comparison of the above description with the definition of normal form of a linear basic grammar (Definition 2.8) it should be clear that we can define a register program to be a regular controlled linear basic grammar in normal form (formally: an m-extended REG-controlled linear basic ONE-grammar in normal form, where m is either OI or IO which makes

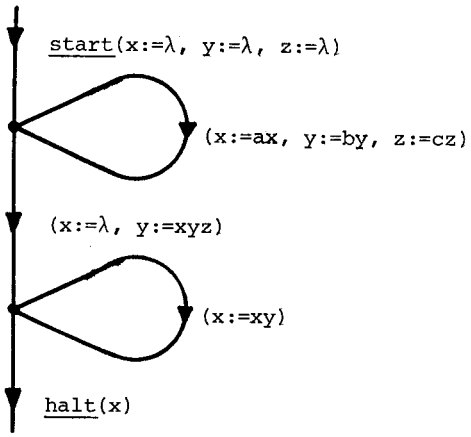


Figure 1.

no difference in this case). Indeed, since the ψ-languages are singletons, there is an obvious correspondence between initial [intermediate, terminal] rules and start instructions [parallel assignments, halt instructions respectively]. Moreover, a derivation in the linear basic grammar is analogous to a computation of the register program. The grammar in normal form corresponding to Fig. 1 has (labeled) rules

p: $S \to E(\lambda,\lambda,\lambda)$

q: $E(x,y,z) \to E(ax,by,cz)$

r: $E(x,y,z) \to E(\lambda,xyz,z)$

s: $E(x,y,z) \to E(xy,y,z)$

t: $E(x,y,z) \to x$

and regular control language pq^*rs^*t.

Hence, by the Normal Form Lemma 2.9, and the Regular Control Lemma 2.10 and the result of [5, 1], the family of languages generable by register programs is equal to EDTOL. Since we are interested in register programs that compute in arbitrary domains, cf. [20], let us now consider the case of an arbitrary algebra A which for each f in a ranked alphabet Δ ($f \in \Delta_n$, $n \geq 0$) has an operation $f_A : A^n \to A$; such an algebra is called a Δ-algebra or algebra of type Δ. Thus we consider nondeterministic computation in algebras which have no predicates (and hence the computation does not involve tests).

The obvious notion of a register program computing in A (and, in fact, in any other Δ-algebra) differs from the case described above by the fact that the right-hand sides of assignments are now expressions (trees) over the operations in Δ and the registers. It should be clear how the computations of such a register program in the Δ-algebra A are defined. Thus the register program defines (generates) a subset of A. On the other hand, since an expression (or tree) is a special kind of string, the register program also generates a tree language over Δ.

As an example consider the register program G with (labeled) instructions ℓ_1 : start(x:=a), ℓ_2 : (x:=f(x,x)), ℓ_3 : halt(x), and the regular control language $\ell_1\ell_2^*\ell_3$. Clearly G generates the language of all binary balanced trees over the ranked alphabet $\Delta = \Delta_0 \cup \Delta_2$ with $\Delta_0 = \{a\}$ and $\Delta_2 = \{f\}$. If A is the Δ-algebra of integers with $a_A = 1$ and f_A is addition, then G defines the set of integers $\{2^n \mid n \geq 0\}$.

We will show (Theorem 4.4) that the subset of A defined by the register program is completely determined by the tree language generated by it. Thus, instead of computing in the algebra A one may as well compute symbolically in the algebra of trees over Δ and then interpret the symbolic output (for a discussion of such "Mezei and Wright like" results see for instance [9, 19, 20, 21]). Consequently we are interested in the family ILBT of "iterative" linear basic tree grammars (defined in [5, 1]; the adjective iterative is added to forbid rules of the form $F(x_1,\ldots,x_n) \rightarrow uG(\ldots)v$ with $uv \neq \lambda$ which provide more power in the tree case [5]). It was shown in [5, 1] that ILBT is equal to the family EDTOLT of EDTOL tree languages (defined in the obvious way, the trees growing in parallel at their leaves).

We will show that these results can be generalized to the extended (controlled) case. An extended register program will be defined as an extended

linear basic tree grammar in normal form. An intermediate rule $E(\underline{x}) \to E(\psi_1(\underline{x}),\ldots,\psi_n(\underline{x}))$ will be interpreted as the parallel assignment $(x_1:=\psi_1(x_1,\ldots,x_n),\ldots,x_n:=\psi_n(x_1,\ldots,x_n))$. Given an algebra A the program can be IO-executed or OI-executed. The IO-execution is the most natural one: the program keeps elements of A in its registers (as in the non-extended case) and interprets the ψ_i as (nondeterministic) function procedures. For these function procedures it is assumed that a Mezei and Wright like result holds; hence it suffices to specify the tree languages generated by them. In the OI-execution the program keeps subsets of A in its registers and interprets the $\psi_i(x_1,\ldots,x_n)$ as operations on subsets; thus we may view the program to be a non-extended program computing in the subset algebra of A with the ψ_i as operations.

We now start the formal part of this section. Before defining the notion of extended register program we recall some terminology concerning trees. As before, PC denotes the finite set consisting of the left parenthesis, the right parenthesis and the comma. For a ranked alphabet Δ, the set of trees over Δ, denoted by T_Δ, is the smallest string language over $\Delta \cup PC$ such that (i) $\Delta_0 \subseteq T_\Delta$ and (ii) if $t_1,\ldots,t_n \in T_\Delta$ and $f \in \Delta_n$, then $f(t_1,\ldots,t_n) \in T_\Delta$. If X is a set of variables (or registers), then $T_\Delta(X)$ denotes $T_{\Delta \cup X}$, where the elements of X have rank 0. A set of trees (over some ranked alphabet) is called a tree language.

Definition 4.1. An (m,Γ,K)-elb grammar $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ is tree generating if $\Sigma = \Delta \cup PC$ for some ranked alphabet Δ and, for each rule $\psi(x_1,\ldots,x_n) \to L$ in P, L is a tree language in $T_\Delta(\{x_1,\ldots,x_n\})$. □

It is easy to see that, for such a tree generating G, $L_m(G)$ is a tree language over Δ.

The corresponding families of tree languages will be denoted by adding I (for "iterative") in front and T (for "tree") at the end of the name in the string case. Thus we have $ILB_{OI}T(\Gamma,K)$ and $ILB_{IO}T(\Gamma,K)$. Note that $ILB_{IO}T(REG,ONE) = ILB_{OI}T(REG,ONE) = ILBT$ [5].

We now define extended register programs.

Definition 4.2. An m-extended Γ-controlled register program with basis K (abbreviated by (m,Γ,K)-extended register program) is an (m,Γ,K)-elb grammar which is tree generating and in normal form. □

When discussing extended register programs, an initial rule $S \to E(\psi_1,\ldots,\psi_n)$ is written as $\underline{start}(x_1:=\psi_1,\ldots,x_n:=\psi_n)$, an intermediate rule $E(\underline{x}) \to E(\psi_1(\underline{x}),\ldots,\psi_n(\underline{x}))$ as the parallel assignment $(x_1:=\psi_1(x_1,\ldots,x_n),\ldots,x_n:=\psi_n(x_1,\ldots,x_n))$, and a terminal rule $E(\underline{x}) \to \psi(\underline{x})$ as $\underline{halt}(\psi(x_1,\ldots,x_n))$. The variables $x_1,\ldots,x_n$ of the program are called registers.

Perhaps it is more appropriate to talk about program schemes rather than programs, because the interpretation is undefined. We use both terms.

In order to define the IO and OI computation of an extended register program in an algebra, we need some additional (algebraic) terminology (cf. [20, 14, 9]). Let A be a Δ-algebra, consisting of a domain (also denoted by A) and an operation $f_A: A^k \to A$ for each $f \in \Delta_k$ $(k \geq 0)$. For each tree $t \in T_\Delta$ we denote by t_A its interpretation as an element of A, thus if $t = f(t_1,\ldots,t_k)$ then $t_A = f_A(t_{1A},\ldots,t_{kA})$. Moreover, if $L \subseteq T_\Delta$ then L_A denotes the subset $\{t_A \mid t \in L\}$ of A. If $X = \{x_1,\ldots,x_n\}$ then, for each $t \in T_\Delta(X)$ we denote by t_A the corresponding "derived operation" $A^n \to A$ where we assume that X has a fixed order). Thus, if $t = x_i$ then t_A is the i-th projection and if $t = f(t_1,\ldots,t_k)$ then t_A is the composition of f_A with $t_{1A},\ldots,t_{kA}$. The powerset $P(A)$ of A is made into a Δ-algebra, the

subset algebra of A, by extending the operations of A element-wise to $P(A)$. Thus, for $f \in \Delta_k$, $f_{P(A)}(A_1,\ldots,A_k) = \{f_A(a_1,\ldots,a_k) \mid a_i \in A_i\}$ where $A_i \subseteq A$. For $L \subseteq T_\Delta(X)$ we denote by $L_{P(A)}$ the "derived operation" on $P(A)$ defined by $L_{P(A)}(A_1,\ldots,A_n) = \cup \{t_{P(A)}(A_1,\ldots,A_n) \mid t \in L\}$.

We now formalize the IO and OI execution of an extended register program, as informally explained before.

Definition 4.3. Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be a (controlled) extended register program with $\Sigma = \Delta \cup PC$ for some ranked alphabet Δ, and $X = \{x_1,\ldots,x_n\}$. Let A be a Δ-algebra.

(i) An IO-configuration is either S, or a sequence $(a_1,\ldots,a_n)$ with $a_i \in A$, or a single element a of A. The relation $\vdash_{IO}$ between IO-configurations is defined as follows.

- If ℓ : start$(x_1:=\psi_1,\ldots,x_n:=\psi_n)$ is in P, $\psi_i \to L_i$ is in P and $t_i \in L_i$ for $1 \le i \le n$, then we write $S \vdash_{IO}^{\ell} (t_{1A},\ldots,t_{nA})$.
- If ℓ : $(x_1:=\psi_1(\underline{x}),\ldots,x_n:=\psi_n(\underline{x}))$ is in P, $\psi_i(\underline{x}) \to L_i$ is in P and $t_i \in L_i$ for $1 \le i \le n$, then we write, for all $a_1,\ldots,a_n \in A$, $(a_1,\ldots,a_n) \vdash_{IO}^{\ell} (t_{1A}(a_1,\ldots,a_n),\ldots,t_{nA}(a_1,\ldots,a_n))$.
- If ℓ : halt$(\psi(\underline{x}))$ is in P, $\psi(\underline{x}) \to L$ is in P and $t \in L$, then we write $(a_1,\ldots,a_n) \vdash_{IO}^{\ell} t_A(a_1,\ldots,a_n)$ for all $a_1,\ldots,a_n \in A$.
- The subset of A IO-generated by G is $S_{IO}(G,A) = \{a \in A \mid S \vdash_{IO}^{\pi} a$ for some $\pi \in M\}$, where $\vdash_{IO}^{\pi}$ is defined in the usual way for $\pi \in \Lambda^*$.

(ii) An OI-configuration is either S, or a sequence $(A_1,\ldots,A_n)$ with $A_i \subseteq A$, or a single subset A' of A. The relation $\vdash_{OI}$ between OI-configurations is defined as follows.

- If ℓ : start$(x_1:=\psi_1,\ldots,x_n:=\psi_n)$ is in P and $\psi_i \to L_i$ is in P, then $S \vdash_{OI}^{\ell} (L_{1A},\ldots,L_{nA})$.
- If ℓ : $(x_1:=\psi_1(\underline{x}),\ldots,x_n:=\psi_n(\underline{x}))$ is in P and $\psi_i(\underline{x}) \to L_i$ is in P, then

$(A_1,\ldots,A_n) \vdash_{OI}^{\ell} (L_{1P(A)}(A_1,\ldots,A_n),\ldots,L_{nP(A)}(A_1,\ldots,A_n))$ for all $A_1,\ldots,A_n \subseteq A$.

- If ℓ : halt($\psi(\underline{x})$) is in P and $\psi(\underline{x}) \to L$ is in P, then for all $A_1,\ldots,A_n \subseteq A$, $(A_1,\ldots,A_n) \vdash_{OI}^{\ell} L_{P(A)}(A_1,\ldots,A_n)$.
- The subset of A OI-generated by G is $S_{OI}(G,A) = \cup \{A' \subseteq A \mid S \vdash_{OI}^{\pi} A'$ for some $\pi \in M\}$. □

The reader should notice that this definition of execution of a register program is completely analogous to the $\vdash$-definition of derivation in an elb-grammar (Definition 2.6). This analogy enables us to obtain the announced Mezei and Wright like result, which says that the subset generated by a register program G is equal to the interpretation $L(G)_A$ of the tree language L(G) generated by it.

Theorem 4.4. Let G be a (controlled) extended register program over the ranked alphabet Δ and let A be a Δ-algebra. Then $S_{IO}(G,A) = L_{IO}(G)_A$ and $S_{OI}(G,A) = L_{OI}(G)_A$.

Proof. Each $\vdash_{IO}$ derivation of the grammar G (Definition 2.6) is turned into an $\vdash_{IO}$ computation of the program G (Definition 4.3) by applying the transformation $t \mapsto t_A$ and in this way all such computations are obtained. This can easily be proved using the well-known fact that substitution of trees is interpreted in A as composition of derived operations (see for instance Proposition 2.5 of [14]). The same is true for the OI-case with the transformation $L \mapsto L_A$ using the analogous fact that (nondeterministic) substitution of tree languages is interpreted in the subset algebra $P(A)$ as composition of derived operations (see Theorem 2.4.1 of [9]). □

It follows from this theorem and the way the ordinary OI-derivation of an elb-grammar has been defined (viz. as OI-derivation in a macro grammar with countably many rules), that the ψ's in an OI-extended register program may also be viewed as symbolic function procedures. From this point of view the OI-execution starts with a symbolic computation on expressions which contain ψ's and, after the application of a terminal rule, calls the function procedures ψ in an OI fashion. We note that the same is not true in the IO-case (cf. the remarks after Lemma 2.7 and the example at the end of Section 2).

Two program(scheme)s are equivalent if they compute the same subset in every algebra. Note that T_Δ is a Δ-algebra, and $S_{IO}(G,T_\Delta) = L_{IO}(G)$ and $S_{OI}(G,T_\Delta) = L_{OI}(G)$. From this and Theorem 4.4 we obtain the following result, which shows that two programschemes are equivalent iff they compute the same subset in the "free interpretation" T_Δ.

Corollary 4.5. Two controlled extended register programs are equivalent iff they generate the same tree language. □

Thus it is natural to consider the family of tree languages generated by (m,Γ,K)-extended register programs, which is equal to $ILB_m T(\Gamma,K)$ (assuming that SYMBOL $\subseteq$ K and Γ is closed under intersection with regular sets), because the Normal Form Lemma 2.9 preserves the tree generating property of the grammar, as the reader may easily check. We now turn to the characterization of this class by iteration grammars.

Definition 4.6. A (controlled) iteration grammar $G = (V,\Sigma,U,I,M,S)$ is tree generating if $V = \Delta \cup PC$ for some ranked alphabet Δ, Σ includes

PC $\cup \bigcup_{n\geq 1} \Delta_n$ and, for each (deterministic or nondeterministic) substitution $\tau_i \in U$, $\tau_i(a)$ is a tree language over Δ for $a \in \Delta_0$, and $\tau_i(a) = \{a\}$ otherwise.□

Intuitively the sentential forms of a tree generating iteration grammar are trees over Δ which grow by rewriting their leaves by trees (in parallel). A derivation is succesful when all leaves of the last sentential form are terminal.

The involved families of tree languages are denoted by $HT(\Gamma,K)$ and $\eta T(\Gamma,K)$. For $K = ONE$, the class $HT(\Gamma,ONE) = \eta T(\Gamma,ONE)$ will also be denoted by (Γ)EDTOLT: the Γ-controlled EDTOL tree languages (cf. [5]).

Since the constructions used in Section 3 to prove the equivalence of extended linear basic grammars and iteration grammars preserve the tree generating property (as may easily be checked), we immediately have the following corollary.

Theorem 4.7. Let Γ and K satisfy all assumptions of the Equivalence Theorem 3.4. Then the family of tree languages generated by (IO,Γ,K)-extended register programs is equal to $ILB_{IO}T(\Gamma,K) = \eta T(\Gamma,K)$, and similarly for OI it is $ILB_{OI}T(\Gamma,K) = HT(\Gamma,K)$. □

Theorems 4.4 and 4.7 together express the main result of this section: extended register programs are characterized by iteration tree grammars and vice versa, with the extended linear basic tree grammars as intermediate concept. Since for register programs the IO-case is the most natural one, this provides additional motivation (of which we were not aware in [4]) for studying deterministic iteration grammars.

In the remaining part of this section we present a few examples of the use of this relationship between extended register programs and iteration grammars. As a first example we show that if Γ is closed under (nondeterministic)

substitution, then $\eta(\Gamma,K)$ is closed under Γ-controlled iterated deterministic substitution.

Theorem 4.8. Let Γ and K satisfy the assumptions of the Equivalence Theorem 3.4; let, moreover, Γ be closed under substitution and K under homomorphism. Then $\eta(\Gamma,\eta(\Gamma,K)) \subseteq \eta(\Gamma,K)$.

Proof. From Theorem 4.7 it follows that $\eta(\Gamma,K)$ equals the family of languages generated by (IO,Γ,K)-extended register programs which compute in the algebra of strings with concatenation as basic operation, i.e. the ψ's in the right-hand sides of assignments represent string languages. Similarly for $\eta(\Gamma,\eta(\Gamma,K))$; note that, by Lemma 4.2 of [4], $\eta(\Gamma,K)$ satisfies the same assumptions as those for K. Intuitively the theorem now follows from the fact that the use of (nonrecursive) function procedures in register programs may be avoided by replacing them by their bodies.

Let $G = (\Phi,\Psi,\Sigma,X,P,S,\Lambda,M)$ be a IO-extended Γ-controlled register program with basis $\eta(\Gamma,K)$, and let $X = \{x_1,\ldots,x_n\}$. It suffices to show that there exists an equivalent IO-extended Γ-controlled register program H with basis K. H is simply obtained from G by replacing calls of function procedures in G by the register programs of $\eta(\Gamma,K)$ which define them, as follows.

Consider an assignment ℓ : $(x_1:=\psi_1(\underline{x}),\ldots,x_n:=\psi_n(\underline{x}))$ of G and let $\psi_i(\underline{x}) \to L_i$ be in P. Since L_i is in $\eta(\Gamma,K)$, it is generated by an (IO,Γ,K)-extended register program G_i. Let M_i be the control language of G_i. Note that the $x_1,\ldots,x_n$ are terminals in G_i; it may therefore be assumed that the registers of G_i are disjoint with $\{x_1,\ldots,x_n\}$. Replace in G_i each instruction $\underline{halt}(\psi(\underline{x}))$ by the assignment $(y_i:=\psi(\underline{x}))$ where y_i is a completely new register. We now, intuitively, change the above assignment of G into the register program $G_1;G_2;\ldots;G_n;(x_1:=y_1,\ldots,x_n:=y_n)$. Note that in this context the

$x_1,\ldots,x_n$ in G_i refer to the actual registers of G. Formally, we define a substitution τ such that $\tau(\ell) = M_1M_2\ldots M_nk$, where k is the label of $(x_1:=y_1,\ldots,x_n:=y_n)$. It is left to the reader to handle the start and halt instructions of G. Now H consists of the instructions of all register programs involved (except G) and has control language $\tau(M)$. It should be clear that program H is equivalent to program G. □

The proof of this theorem is an example of the application of "register programming" to iteration grammars. Clearly, a similar result holds for trees. For the OI-case the analogous property $H(\Gamma,H(\Gamma,K)) \subseteq H(\Gamma,K)$ is more complicated to prove by register programming than by the usual construction on iteration grammars (cf. the proof that $H(H(\Gamma,K)) \subseteq H(\Gamma,K)$ in [3]). This case can therefore serve as an example of the application of iteration grammar techniques to register programs (which compute in subset algebras): it shows that the use of nonrecursive function procedures is allowed for such programs.

As a second and last example we compare a few classes of recursive program schemes. First, consider the family of CF-controlled register programs (with basis ONE), i.e. non-extended register programs (computing in an arbitrary algebra) with context-free control. It should be clear that the context-free control stands for the use of parameterless recursive procedures (with global registers). By Theorems 4.4 and 4.7 these programs are characterized by the class (CF)EDTOLT which we also denote by (CF)ILBT: the class of context-free controlled (iterative) linear basic tree languages. An (informal) example of such a program is $G = (x:=a_1,y:=a_2);$ call S; halt$(h(x,y))$ over the ranked alphabet Δ with $\Delta_0 = \{a_1,a_2\}$, $\Delta_1 = \{f_1,g_1,f_2,g_2\}$ and $\Delta_2 = \{h\}$, where S is the recursive

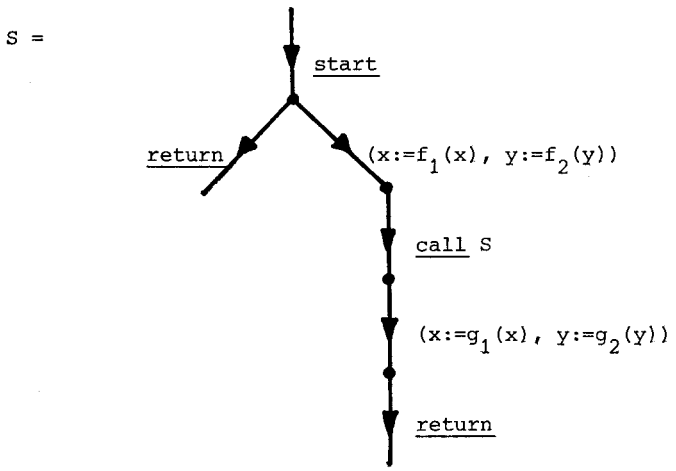


Figure 2.

procedure given in Fig. 2. G generates the tree language $\{h(g_1^n f_1^n(a_1), g_2^n f_2^n(a_2)) \mid n \geq 0\}$.

The other families of program schemes which we consider are the IO and OI nondeterministic polyadic recursion schemes (also without tests), as discussed for instance in [9]. It is shown in [9] that these schemes are characterized (in the Mezei and Wright way) by the families IOT and OIT respectively, where IOT is the family of tree languages generated by IO-macro tree grammars (defined in the obvious way), and similarly for OI. Note that EDTOLT (= ILBT) is included in both IOT and OIT. Also the family RECOG of recognizable tree languages [20] is included in both IOT and OIT: they represent the parameterless recursion schemes which define the "equational" subsets of an algebra [20]. Observe that the IO-extended (regular controlled) register programs with basis RECOG, which in any algebra compute from the equational subsets, generate the family $\eta T(REG,RECOG)$, which is closely related to the family $\eta(CF)$, investigated in [4].

We want to show that the three families of program schemes represented by (CF)ILBT, IOT and OIT are incomparable (with respect to program scheme equivalence). By Corollary 4.5 and the results of [9] it suffices to show that (CF)ILBT, IOT and OIT are incomparable families of tree languages.

Since the families IO and OI of macro languages are incomparable [12], so are IOT and OIT.

The tree language $\{h(g_1^n f_1^n(a_1), g_2^n f_2^n(a_2)) \mid n \geq 0\}$ is in (CF)ILBT as shown by the above example (note that the control is even linear), but is not in OIT by the copying theorem of [2]: if $\{h(t,t) \mid t \in L\} \in$ OIT, then $L \in$ ILBT (but tree languages in ILBT have regular path languages). It is straightforward to show the similar result that if $\{h(t,\bar{t}) \mid t \in L\} \in$ IOT, then $L \in$ ILBT (where $\bar{t}$ is a copy of t over a disjoint alphabet). This shows that the above tree language is not in IOT as well. We observe here that the situation is quite different for the usual deterministic recursive register programs and deterministic recursion schemes (both with tests). In fact, in the deterministic case, every recursive register program is equivalent to some recursion scheme (see e.g. [15]). Thus, the above example shows that nondeterminism destroys this obvious relationship.

Also,in the deterministic case, recursion schemes are more powerful than register programs [22]. We now show that this remains true in the nondeterministic case. Let Δ be the ranked alphabet with $\Delta_0 = \{a\}$, $\Delta_1 = \{\ell,r\}$ and $\Delta_2 = \{f\}$. Then the tree language T_Δ can be defined by a recursion scheme (it is even in RECOG), but not by any Γ-controlled register program. To see this consider a tree over Δ consisting of an upper part which is a binary balanced tree of some height n, and a lower part which is obtained by attaching, in some way, a different (monadic) tree over $\{\ell,r,a\}$ at each leaf of the upper part. It was shown in [22], see also [15], that to compute

such a tree one needs n + 1 registers (note that the different monadic subtrees do not allow the program to use copying when computing the upper part). Thus, independent of the control, a program with a bounded number of registers cannot compute a tree language which contains all such trees. For a similar proof that RECOG is not included in ILBT, see [1].

The above arguments show that (CF)EDTOLT, IOT and OIT are incomparable classes of tree languages (or, alternatively, programs), each incorporating a different kind of (nondeterministic) recursion.

Let us end by briefly considering the string case. For the interested reader we note here that the "language of cuts", used in [10] to prove the existence of a language in IO ∩ OI not in ETOL, is related to the above as follows. The example of a recursion scheme used in [22] is F(x) = if p(x) then x else f(F(ℓ(x)), F(r(x))), where p is a test. Turning this into a nondeterministic recursion scheme without test we obtain the (IO and OI) macro tree grammar with rules S → F(a), F(x) → f(F(ℓ(x)), F(r(x))) and F(x) → x. The generated tree language contains all trees of the type mentioned above, where to each leaf of the upper part a code is attached of the path leading to this leaf (ℓ = left and r = right). This tree language is therefore not computable by a (controlled) register program. The language of cuts was (approximately) defined by the macro grammar with rules S → F(a), F(x) → F(ℓx)F(rx) and F(x) → x. The proof in [10] that this language cannot be computed by a (regular controlled) register program uses a pumping lemma for EDTOL. We conjecture that this language is not even in (CF)EDTOL.

We also note that another version of T_Δ (viz. the Dyck set D_2 over two kinds of parentheses) is even not computable by a register program computing on strings, assuming of course that the control is less than CF [6, 8]. Finally we remark that $\{w\#\bar{w}\#\bar{\bar{w}} \mid w \in D_2\}$ is in (CF)EDTOL, but not in IO or OI, which can be proved using methods of [12], cf. [11], and the previous fact.

Conclusion

Extended linear basic grammars have been defined in a way such that, if the "extension family" K is chosen as a family of macro languages, then they still generate macro languages. Since we showed that extended linear basic grammars correspond to iteration grammars, we obtained a tool to investigate in more detail the closure properties of macro languages with respect to iterated substitution. On the other hand this equivalence can be used - as in [10] for the finite case - to obtain a machine model (different from the one in [24]) characterizing full hyper-AFL's.

A register program model for extended linear basic grammars has been studied. Due to its intuitive appeal this model might be a very promising alternative to iteration grammars.

References

1. A. Arnold, M. Dauchet, Transductions de forêts reconnaissables monadiques; forêts corégulières, Rev. Française Automat. Informat. Rech. Opérat. IT 10 (1976) 5-28.
2. A. Arnold, M. Dauchet, Un théorème de duplication pour les forêts algébriques, J. Comp. System Sci. 13 (1976) 223-244.
3. P.R.J. Asveld, Controlled iteration grammars and full hyper-AFL's, Inform. Contr. 34 (1977) 248-269.
4. P.R.J. Asveld, J. Engelfriet, Iterated deterministic substitution, Acta Informatica 8 (1977) 285-302.
5. P.J. Downey, Formal languages and recursion schemes, Ph. D. Thesis TR 16-74, 1974, Harvard University, Cambridge, Mass.
6. A. Ehrenfeucht, G. Rozenberg, On some context-free languages that are not deterministic ETOL languages, Rev. Française Automat. Informat. Rech. Opérat. IT 11 (1977) 273-291.

7. J. Engelfriet, Simple program schemes and formal languages, Lecture Notes in Computer Science 20 (1974), Springer, Berlin-Heidelberg-New York.

8. J. Engelfriet, G. Rozenberg, G. Slutzki, Tree transducers, L systems and two-way machines, TW-memorandum 187 (1977), Twente University of Technology, Enschede, The Netherlands.

9. J. Engelfriet, E.M. Schmidt, IO and OI; Part I, J. Comp. System Sci. 15 (1977) 328-353; Part II, J. Comp. System Sci. (to appear).

10. J. Engelfriet, E.M. Schmidt, J. van Leeuwen, Stack machines and classes of nonnested macro-languages, Report RUU-CS-77-2 (1977), University of Utrecht, The Netherlands.

11. J. Engelfriet, S.Skyum, Copying theorems, Inform. Processing Letters 4 (1976) 157-161.

12. M.J. Fischer, Grammars with macro-like productions, Ph. D. Thesis (1968), Harvard University, Cambridge, Mass.

13. S.J. Garland, D.C. Luckham, Program schemes, recursion schemes, and formal languages, J. Comp. System Sci. 7 (1973), 119-160.

14. J.A. Goguen, J.W. Thatcher, E.G. Wagner, J.B. Wright, Initial algebra semantics and continuous algebras, J. Ass. Comp. Mach. 24 (1977) 68-95.

15. S.A. Greibach, Theory of program structures: schemes, semantics, verification, Lecture Notes in Computer Science 36 (1975), Springer, Berlin-Heidelberg-New York.

16. G.T. Herman, A biologically motivated extension of ALGOL-like languages, Inform. Contr. 22 (1973) 487-502.

17. G.T. Herman, G. Rozenberg, Developmental systems and languages, (1975) North-Holland, Amsterdam.

18. J.E. Hopcroft, J.D. Ullman, Formal languages and their relation to automata, (1969) Addison-Wesley, Reading, Mass.

19. W.E. Howden, Lindenmayer grammars and symbolic testing, Inform. Processing Letters 7 (1978) 36-39.

20. J. Mezei, J.B. Wright, Algebraic automata and context-free sets, Inform. Contr. 11 (1967) 3-29.

21. M. Nivat, On the interpretation of recursive program schemes, Symposia Matematica - Atti del convegno d'Informatica teorica, Roma (1973).

22. M.S. Paterson, C.E. Hewitt, Comparative schematology, pp. 119-128 in Record of Project MAC Conference on concurrent systems and parallel computation (1970), Ass. Comp. Mach., New York.

23. G. Rozenberg, Extension of tabled OL-systems and languages, Internat. J. Comp. Inform. Sci. 2 (1973) 311-336.

24. G. Rozenberg, D.Vermeir, Acceptors for iteration languages, pp. 460-464 in J. Gruska (Ed.) "Mathematical Foundations of Computer Science 1977", Lecture Notes in Computer Science 53 (1977), Springer, Berlin-Heidelberg-New York.

25. A. Salomaa, Formal languages, (1973), Academic Press, New York.

26. A. Salomaa, Macros, iterated substitution and Lindenmayer-AFLs, DAIMI PB-18 (1973), University of Aarhus, Denmark. An abstract appeared in: "L systems", Lecture Notes in Computer Science 15 (1974) 250-253, Springer, Berlin-Heidelberg-New York.

27. J. van Leeuwen, F-iteration languages, Memorandum (1973), University of California, Berkeley.

28. J. van Leeuwen, A generalization of Parikh's theorem in formal language theory, pp. 17-26 in J. Loeckx (Ed.), "Automata, Languages and Programming, 2nd Colloquium" Lecture Notes in Computer Science 14 (1974) Springer, Berlin-Heidelberg-New York.

P A R T I I

FIXED POINTS AND CANONICAL FORMS

Chapter 5

Extensions of Language Families and Canonical Forms for Full AFL-structures

Reprinted from *TW-memorandum No. 167 (May 1977)*

Abstract

We consider the following ways of extending a family of languages K to an "enriched" family X(K) : (i) hyper-algebraic extension (X = H) based on iterated parallel substitution, (ii) algebraic extension (X = A) obtained by nested iterated substitution, (iii) rational extension (X = R) achieved by not self-embedding nested iterated substitution, and (iv) a few subrational extensions (X = M, S, P, F, C) based on several kinds of substitution. We introduce full X-AFL's, i.e. non-trivial families closed under finite substitution, intersection with regular sets, and under X, which turn out to be equivalent to well-known AFL-structures such as full hyper-AFL (X = H), super-AFL (A), substitution-closed AFL (R), semi-AFL (S), etc. Then we establish Canonical Forms for the smallest full X-AFL $\hat{\mathcal{X}}(K)$ including K, i.e. we decompose the operator $\hat{\mathcal{X}}$ into simpler operators. Using Canonical Forms for full X-AFL's we obtain expressions for the smallest full X-AFL including the result of substituting a family of languages into another family.

1. INTRODUCTION

In studying closure properties of language families major progress was made since 1969 when Ginsburg, Greibach & Hopcroft [10] introduced the concept of Abstract Family of Languages (AFL) denoting any family closed under the regular operations (union, concatenation and Kleene +), homomorphism, inverse homomorphism and intersection with regular sets. Using this concept we consider language families as algebras which enables us to deal with closure properties in a more abstract fashion.

As usual in algebra, structures with more restrictive and, on the other hand more general properties have been considered. Ginsburg & Spanier [11] investigated substitution-closed AFL's generalizing a well-known property of the regular languages, and Greibach [14, 15] studied AFL's closed under nested iterated substitution (super-AFL's) featuring a property of the context-free languages. In the theory of parallel rewriting (in particular in the case of ETOL languages; cf. [21]) AFL's closed under iterated parallel substitution (hyper-AFL's) were intensively investigated [24, 23, 4, 2]. On the other hand many theorems originally proved for AFL's also hold for weaker structures like semi-AFL (inspired by the linear context-free languages) and trio. For the main results on trio, (semi-)AFL and substitution-closed AFL we refer to a survey by Ginsburg [9].

Van Leeuwen [24] and Salomaa [23] originally introduced (full) hyper-AFL's and a related rewriting system called K-iteration grammar, where K refers to an arbitrary family of languages. A K-iteration grammar is essentially an ETOL system [21] in which each table (i.e. finite substitution) has been replaced by an arbitrary K-substitution. The family of languages generated by K-iteration grammars is called the hyper-algebraic extension of the family K and will be denoted by H(K). According to [24, 23,2] a nontrivial family K is a full hyper-AFL iff (i) K is closed under finite substitution, (ii) K is closed under intersection with regular sets, and (iii) K is hyper-algebraically closed, i.e. H(K) = K. In [2] it was shown that the smallest full hyper-AFL $\hat{\mathfrak{A}}(K)$ containing the family K equals the hyper-algebraic extension of the smallest pre-quasoid $\Pi(K)$ (i.e. the smallest nontrivial family satisfying (i) and (ii)) containing K, i.e. $\hat{\mathfrak{A}}(K) = H\Pi(K)$. This "Canonical Form Theorem" means that the operator $\hat{\mathfrak{A}}$ can be decomposed into a single product (composition) of the simpler operators H and Π.

In this paper we investigate similar ways of connecting other well-known AFL-structures with rewriting systems (like restricted kinds of K-iteration grammars such as e.g. context-free K-grammars introduced in [25]), in such a way that

(A) the corresponding extension X (i.e. the family of languages generated by that particular kind of restricted iteration grammar) which may be considered as an operator on families of languages, characterizes the original AFL-structure by means of the conditions: (1) containment of a nontrivial language, (2) closure under finite substitution, (3) closure under intersection with regular sets, and (4) closure under the operator X. A family satisfying (1) - (4) will be called a full X-AFL. Thus K is a full X-AFL when $\Pi(K) = K$ and $X(K) = K$.

(B) a Canonical Form Theorem for $\hat{\mathcal{X}}(K)$ (i.e. the smallest full X-AFL containing K) holds or, equivalently, it is possible to factorize the operator $\hat{\mathcal{X}}$ into a single product of X and Π: $\hat{\mathcal{X}}(K) = X\Pi(K)$.

The particular types of AFL-structures that will be considered in this paper are full trio, semi-AFL, pseudo-AFL (i.e. full semi-AFL closed under concatenation), Kleene-AFL (i.e. full trio closed under Kleene *), AFL [9], substitution-closed AFL [11], super-AFL [14, 15], hyper(1)-AFL [24, 4, 8] and hyper-AFL [24, 23, 4, 2].

From (A) it should be clear that $\hat{\mathcal{X}}(K) = \bigcup\{\Omega(K) \mid \Omega \in \{X,\Pi\}^*\}$, or in a more algebraic notation, that $\hat{\mathcal{X}}(K) = \{X,\Pi\}^*(K)$. Thus the Canonical Forms as mentioned in (B) imply that instead of the infinite set of operators $\{X,\Pi\}^*$ we only need to consider the application of one single operator $X\Pi$. Therefore Canonical Forms can be succesfully used in proving certain families to be full X-AFL's and in establishing properties of $\hat{\mathcal{X}}(K)$.

All extensions in this paper are obtained either "directly" by natural generalizations of basic families (like ETOL, EOL, context-free and regular languages) or "indirectly" by fixing parameters in extensions already defined in a "direct" way. Investigating extensions of language families instead of basic families enables us to uncover structural features and arguments for the rewriting mechanisms involved, rather than strictly combinatorial proofs usually given in dealing with basic families. This rather algebraic approach to language theory does not only provide a unifying framework for investigating closure properties (to which the present paper is mainly restricted) but has also succesfully been applied in other areas of formal language theory (cf. e.g. [25, 26, 27]).

This approach also emphasizes again the principal role played by the concept of pre-quasoid (or equivalently, the notion of nondeterministic generalized sequential machine with accepting states; cf. [18, Chapter 9]) in dealing with families closed under several kinds of (iterated) substitution [2, 8, 12, 15, 19, 23, 24, 26].

This paper is organized into five sections of which this introductory section is the first. Section 2 contains preliminaries from formal language theory, definitions of the basic rewriting systems, the corresponding extensions and full X-AFL's. We conclude section 2 with proving basic properties of these extensions and with a few examples. In section 3 we establish Canonical Forms for full hyper-AFL, hyper(1)-AFL, algebraic AFL (i.e. super-AFL) and rational AFL (i.e.substitution-closed AFL). Canonical Forms for the other cases (i.e. full trio, semi-AFL, pseudo-AFL, Kleene-AFL, AFL) are proved in section 4. The results of sections 3 and 4 are applied in section 5 to the family obtained by substituting languages from a family into languages from another family. Section 5 also contains a few generalizations of results of section 4 and an example of the application of Canonical Forms and related theorems in proving certain families to be particular kinds of full AFL-structures.

2. DEFINITIONS AND BASIC PROPERTIES.

For terminology and basic results in formal language theory we refer to Salomaa [22] or to Hopcroft & Ullman [18]. Moreover the reader is assumed to be familiar with standard concepts from the theory of parallel rewriting (cf. Herman & Rozenberg [17]) and from AFL-theory (cf. Ginsburg [9]).

Let Σ_ω be an arbitrary but fixed countably infinite set of symbols. A family (of languages) is a collection of languages over finite subsets (called alphabets) of Σ_ω.

Let λ denote the empty word. A language L is nontrivial if L is nonempty and $L \neq \{\lambda\}$. A family is called nontrivial when it contains at least one nontrivial language.

Let ONE be the family consisting of all singletons, i.e.

ONE = $\{\{w\} \mid w \in \Sigma_\omega^*\}$, and let SYMBOL be the subfamily containing all singletons of length 1, i.e. SYMBOL = $\{\{\alpha\} \mid \alpha \in \Sigma_\omega\}$. Moreover we define the families ALPHA and STAR by ALPHA = $\{\{\alpha_1,\ldots,\alpha_n\} \mid n \geq 1, \alpha_1,\ldots,\alpha_n \in \Sigma_\omega\}$ and STAR = $\{\{\alpha^*\} \mid \alpha \in \Sigma_\omega\}$ respectively. We denote the families of finite, regular (rational), context-free (algebraic), OL-, EOL-, TOL-, ETOL- and indexed [1] languages by FIN, REG, CF, OL, EOL, TOL, ETOL and INDEX respectively.

Let K be a family. A K-substitution τ is a function on an alphabet V, such that for each α in V $\tau(\alpha)$ is a language in K. The function τ is extended in the usual way to words by $\tau(\lambda) = \{\lambda\}$, $\tau(\alpha_1\ldots\alpha_n) = \tau(\alpha_1)\ldots\tau(\alpha_n)$ for each $\alpha_1\ldots\alpha_n \in V^+$, and to languages by $\tau(L) = \bigcup\{\tau(w) \mid w \in L\}$ for each $L \subseteq V^*$. A K-substitution $\tau : V \to K$ with $\tau(\alpha) \subseteq V^*$ for each α in V is called a K-substitution over V.

A ONE-substitution is usually called a homomorphism, and in case K equals FIN or REG τ is called a finite or regular substitution respectively.

With K_1/K_2 we denote the family obtained by all K_1-substitutions applied to K_2-languages, i.e. $K_1/K_2 = \{\tau(L) \mid \tau$ is a K_1-substitution and $L \in K_2\}$. This binary operation on families is neither commutative, nor associative i.e. neither $K_1/K_2 = K_2/K_1$, nor $K_1/(K_2/K_3) = (K_1/K_2)/K_3$ hold in general [11]. But the inclusion $K_1/(K_2/K_3) \subseteq (K_1/K_2)/K_3$ does hold [11], whereas equality holds whenever K_2 is closed under isomorphism (i.e. renaming of symbols)[11]. In the sequel the operator / will be often applied to families closed under isomorphism and consequently parentheses in expressions like $K_1/K_2/K_3$ will be omitted.

A family K_2 is closed under K_1-substitution if $K_1/K_2 \subseteq K_2$. The closure of K_2 under K_1-substitution is the smallest family containing K_2 and closed under K_1-substitution. The family K_1 is closed under substitution into the

family K_2 if $K_1/K_2 \subseteq K_1$. The closure of K_1 under substitution into K_2 is the smallest family containing K_1 and closed under substitution into K_2. A family K is substitution closed when $K/K \subseteq K$. The substitution closure of K is the smallest substitution-closed family K_∞ containing K.

Let $[K_1/]^0K_2 = K_2$ and by induction $[K_1/]^{n+1}K_2 = K_1/(\bigcup_{i=0}^{n}[K_1/]^iK_2)$ $n \geq 0$ and let $[K_1/]^*K_2 = \bigcup_{n=0}^{\infty}[K_1/]^nK_2$. Similarly let $K_1[/K_2]^0 = K_1$ and by induction $K_1[/K_2]^{n+1} = (\bigcup_{i=0}^{n} K_1[/K_2]^i)/K_2$, $n \geq 0$ and let $K_1[/K_2]^* = \bigcup_{n=0}^{\infty} K_1[/K_2]^n$.

Lemma 2.1. (cf. [11]).

(1) $[K_1/]^*K_2$ is the closure of K_2 under K_1-substitution.

(2) $K_1[/K_2]^*$ is the closure of K_1 under substitution into K_2.

(3) $K_\infty = K[/K]^*$.

(4) If $K \subseteq K/K$, then $K_\infty = [K/]^*K$.

We define the following operators on families of languages:

$\Theta(K) = ONE/K$, $\Phi(K) = FIN/K$, and $\Delta(K) = \{L \cap R \mid L \in K \text{ and } R \in REG\}$.

The notion of pre-quasoid was defined in [2] as a slightly weaker variant of the quasoid introduced by Van Leeuwen [24].

Definition. A family K is a pre-quasoid if

(i) K is nontrivial

(ii) K is closed under finite substitution, i.e. $\Phi(K) \subseteq K$

(iii) K is closed under intersection with regular languages, i.e. $\Delta(K) \subseteq K$.

A quasoid is a pre-quasoid containing at least one infinite language.

Each (pre-)quasoid contains all regular (finite) languages. Consequently REG(respectively FIN) is the smallest (pre-)quasoid, and FIN is the only

pre-quasoid which is not a quasoid [2]. For the smallest pre-quasoid $\Pi(K)$ containing K, we have

Lemma 2.2. If K is a nontrivial family, then $\Pi(K) = \Theta\Delta\Phi(K)$.

Proof: It is well-known that a nontrivial family is a pre-quasoid iff it is closed under mappings induced by a-NGSM's (i.e. nondeterministic generalized sequential machines with accepting states; cf. [18] for definitions and details). Moreover each a-NGSM mapping T can be written as $T(L) = h(f(L) \cap R)$, where h is a homomorphism, R is a regular set and f is a finite substitution [18]. Since a-NGSM mappings are closed under composition [7] we have $\Pi(K) = \bigcup_{n=0}^{\infty}[\Theta\Delta\Phi]^n(K) = \Theta\Delta\Phi(K)$. □

The relation between substitution and the operator Δ is given by the following result established by Ginsburg & Spanier [11].

Lemma 2.3.

(1) $\Delta(K_1/K_2) \subseteq \Delta(K_1)/\Delta\Phi(K_2)$

(2) $\Delta(K_1)/\Delta(K_2) \subseteq \Theta\Delta((FIN \cup K_1)/\Phi(K_2))$

The family K is closed under iterated (parallel) substitution if for all finite sets U of K-substitutions and for all L in K : $U^*(L) = \bigcup\{\tau_1 \ldots \tau_n(L) \mid n \geq 0 ; \tau_1, \ldots, \tau_n \in U\}$ is in K. The family K is closed under single iterated (parallel) substitution, when $U^*(L)$ is in K for each L in K and for each singleton $U = \{\tau\}$, i.e. when $\bigcup\{\tau^n(L) \mid n \geq 0\} \in K$. A K-substitution τ over V is called nested if for each α in V : $\alpha \in \tau(\alpha)$. If for each L in a family K and each nested K-substitution τ, $\bigcup\{\tau^n(L) \mid n \geq 0\}$ is in K, then K is said to be closed under nested iterated substitution.

A full trio is a nontrivial family closed under homomorphism, inverse homomorphism and intersection with regular sets. A full semi-AFL is a full trio closed under union. We call a full semi-AFL closed under concatenation a full pseudo-AFL. Thus a full AFL is a full pseudo-AFL closed under Kleene *. We call a full trio closed under Kleene * a full Kleene-AFL. In [9] it has been shown that each full Kleene-AFL closed under either union or concatenation is a full AFL. Similarly it is straightforward to show that each full trio closed under concatenation is a full pseudo-AFL. A full substitution-closed AFL (full super-AFL, full hyper(1)-AFL, full hyper-AFL) is a full AFL closed under substitution (nested iterated substitution, single iterated substitution, iterated substitution respectively). Let $\hat{\mathcal{M}}(K)$, $\hat{\mathcal{C}}(K)$, $\hat{\mathcal{G}}(K)$, $\hat{\mathcal{P}}(K)$, $\hat{\mathcal{F}}(K)$, $\hat{\mathcal{R}}(K)$, $\hat{\mathcal{A}}(K)$, $\hat{\mathcal{H}}_1(K)$ and $\hat{\mathcal{H}}(K)$ denote the least full trio, Kleene-AFL, semi-AFL, pseudo-AFL, AFL, substitution-closed AFL, super-AFL, hyper(1)-AFL and hyper-AFL containing K respectively. In sections 3 and 4 we will show that all these AFL-structures are different.

Ginsburg & Spanier [11] proved the following

Lemma 2.4. If K is a quasoid, then K_∞ is a full substitution-closed AFL.

We now come to the main definitions. They paraphrase and slightly generalize the original definitions introduced by Van Leeuwen [24, 25] and Salomaa [23] (cf. [2]), in order to fit standard AFL-structures in the framework of family extensions (cf. sections 3 and 4).

Definition. Let K_1, K_2 and K_3 be families. A language L is called hyper-algebraic over (K_1,K_2,K_3) when there exist (i) an alphabet V, (ii) a terminal alphabet Σ, (iii) an initial language $L_0 \subseteq V^*$ in K_1,

(iv) an initial K_2-substitution τ_0 over V, (v) a finite set U of K_3-substitutions over V such that $L = U^*(\tau_0(L_0)) \cap \Sigma^*$. If each τ in U is nested, then L is called algebraic over (K_1,K_2,K_3), whereas such an algebraic L is called rational over (K_1,K_2,K_3) when U is not self-embedding, i.e. if for all u in U^+ and for all α in V the implication:

$w_1\alpha w_2 \in u(\alpha) \Longrightarrow (w_1 = \lambda \text{ or } w_2 = \lambda)$

holds, where $w_1,w_2 \in V^*$. The hyper-algebraic (algebraic, rational) extension $\underline{H}(K_1,K_2,K_3)$ ($\underline{A}(K_1,K_2,K_3)$, $\underline{R}(K_1,K_2,K_3)$) of (K_1,K_2,K_3) is the family of all languages hyper-algebraic (algebraic, rational) over (K_1,K_2,K_3).

We are mainly interested in several less general extensions which are easily obtained as particular cases.

Definition. Let K be a family. The hyper-algebraic, algebraic and rational extension of K are respectively H(K) = $\underline{H}$(SYMBOL, SYMBOL, K), A(K) = $\underline{A}$(SYMBOL, SYMBOL, K) and R(K) = $\underline{R}$(SYMBOL, SYMBOL, K).

Let $K_1 = K_2$ = SYMBOL and K_3 = K. Let V,Σ,U,τ_0 and L_0 be as in the previous definition. The construct $G = (V,\Sigma,U,S)$, where $\{S\} = \tau_0(L_0)$ is called a K-iteration grammar, if U is a finite set of K-substitutions over V. We call G a context-free K-grammar (regular K-grammar) if each τ in U is a nested K-substitution over V (and U is not self-embedding). The grammar G generates a language $L(G) = U^*(S) \cap \Sigma^*$ which is a member of of H(K) (A(K) or R(K) respectively).

The m-restricted extensions $H_m(K)$, $A_m(K)$ and $R_m(K)$ are the subfamilies of H(K), A(K) and R(K) respectively, generated by grammars containing at most m K-substitutions in U ($m \geq 1$).

Another type of less general extensions are the following, which we call the subrational extensions of K:

$M(K) = \underline{R}(SYMBOL, K, FIN)$

$C(K) = \underline{R}(STAR, K, FIN)$

$S(K) = \underline{R}(ALPHA, K, FIN)$

$P(K) = \underline{R}(FIN, K, FIN)$

$F(K) = \underline{R}(REG, K, FIN)$

We call a family K σ-simple when (i) K contains a SYMBOL-language and (ii) K is closed under isomorphism ("renaming of symbols"). A σ-simple family closed under union with SYMBOL-languages is called α-simple. Clearly SYMBOL (ALPHA) is the smallest σ-simple (α-simple) family and it is contained in each σ-simple (α-simple) family. Obviously each pre-quasoid is σ-simple. Consider a nontrivial language $L \subseteq \Sigma^*$ from a pre-quasoid K. Let f be a finite substitution and h a homomorphism defined by $f(a) = \{a,b,e\}$ for each a in Σ, $e \notin \Sigma$ and $h(e) = \lambda$, $h(\alpha) = \alpha$ for $\alpha \in \Sigma \cup \{b\}$. Then $L \cup \{b\} = h(f(L) \cap (\Sigma^* \cup be^*))$ is in K. Hence each pre-quasoid is α-simple.

Our first result deals with the relation between extensions and the corresponding m-restricted variants.

Theorem 2.5.

(1) If K is σ-simple, then $H_2(K) = H_m(K) = H(K)$ for each $m \geq 2$.

(2) If K is α-simple, then $A_1(K) = A_m(K) = A(K)$ and $R_1(K) = R_m(K) = R(K)$ for each $m \geq 1$.

Proof: (1) was already established in [2]. Thus it remains to show that for each $m \geq 1$ $A(K) \subseteq A_m(K) \subseteq A_1(K)$ and similarly for R, since the converse inclusions are obvious.

Let $G = (V,\Sigma,U,S)$ be a context-free K-grammar with $U = \{\tau_1,\ldots,\tau_m\}$ for some $m \geq 2$. Define for each k with $1 \leq k \leq m$ an isomorphism $\phi_k(\alpha) = \alpha_k$ (α in V; all α_k's are new symbols) and extend these isomorphisms in the usual way to words and to languages. Define a new alphabet $V_0 = V \cup \{\phi_k(\alpha) \mid \alpha \in V,\ 1 \leq k \leq m\}$. Consider the context-free K-grammar $H = (V_0,\Sigma,\tau,S)$ where the nested K-substitution τ over V_0 is defined by

$\tau(\alpha) = \{\alpha,\alpha_1\}$ $\qquad \alpha \in V$

$\tau(\alpha_k) = \{\alpha_k\} \cup C_k \cup \tau_k(\alpha)$, $\qquad$ with $C_k = \emptyset$ iff $k = m$ and $C_k = \{\alpha_{k+1}\}$ iff $1 \leq k \leq m - 1$.

The basic idea of the simulation of G by H is the following: each occurrence of each symbol β in V_0 may be object to the following replacements (in an "asynchronous way"): (i) changing into α_{k+1} (if $\beta = \alpha_k$, $1 \leq k \leq m - 1$) or α_1 (if $\beta = \alpha$), i.e. from α we can reach α_j for each j ($1 \leq j \leq m$), (ii) substituting $\tau_j(\alpha)$ into that particular instance of $\beta = \alpha_j$, i.e. we simulate the application of τ_j on that occurrence of α_j while the subscript j is removed.

By this construction we obtain $L(G) = L(H)$ and hence $A(K) \subseteq A_m(K) \subseteq A_1(K)$ for each $m \geq 1$. Since the construction preserves the not self-embedding property of the grammar, a similar conclusion holds in the rational case. □

Note that 2.5.(2) is obvious, whenever K is closed under union. In the hyper-algebraic case 2.5.(1) is the best possible result, i.e. in general one cannot reduce the number of substitutions in a K-iteration grammar to one [21, 8].

By relating subrational extensions to (finitely repeated) substitutions we obtain a considerable simplification.

Theorem 2.6. $\underline{R}(K_1,K,FIN) = REG/(K/K_1)$ and consequently

$M(K) = REG/K$, $C(K) = REG/(K/STAR)$,

$S(K) = REG/(K/ALPHA)$, $P(K) = REG/(K/FIN)$,

$F(K) = REG/(K/REG)$, whereas the parentheses can be omitted whenever K is closed under isomorphism.

Proof: Suppose $L \in REG/(K/K_1)$ and let $L = \rho\tau_0(L_0)$ where (1) $L \subseteq \Delta^*$ is in K_1, (2) $\tau_0 : \Delta \to K$ is a K-substitution with $\tau(\alpha) \subseteq B^*$ for each α in Δ, and (3) $\rho : B \to REG$ is a regular substitution with $\rho(\beta) \subseteq \Sigma_\beta^*$ $(\beta \in B)$ and $\Sigma = \bigcup\{\Sigma_\beta \mid \beta \in B\}$. For each $\beta \in B$ let $(V_\beta,\Sigma_\beta,P_\beta,\beta)$ be an ordinary regular grammar generating $\rho(\beta)$ such that all nonterminal alphabets $V_\beta - \Sigma_\beta$ are mutually disjoint. Let $U = \{\tau\}$ where τ is a substitution over $\bigcup\{V_\beta \mid \beta \in B\} \cup \Delta$ defined by

$\tau(\gamma) = \{\gamma\} \cup \{w \mid (\gamma,w) \in P_\beta\} \in FIN$ iff $\gamma \in V_\beta - \Sigma_\beta$ $(\beta \in B)$

$\tau(\gamma) = \{\gamma\}$ otherwise

Clearly U is not self-embedding and we have $L = \rho\tau_0(L_0) = U^*(\tau_0(L_0)) \cap \Sigma^* \in \underline{R}(K_1,K,FIN)$.

Conversely, let $L \in \underline{R}(K_1,K,FIN)$, i.e. there exist alphabets V and Σ, a language L_0 in K_1, a K-substitution τ_0 over V, and a finite set U of finite substitutions over V, such that $L = U^*(\tau_0(L_0)) \cap \Sigma^*$. Since FIN satisfies the conditions required in 2.5.(2) we can (by a similar argument as in the proof of 2.5.(2)) restrict our attention to the case $U = \{\tau\}$. Consider for each α in V the grammar $G_\alpha = (V,\Sigma,\tau,\alpha)$. Since each G_α is not self-embedding and τ is a finite substitution $L(G_\alpha) = (\bigcup\{\tau^n(\alpha) \mid n \geq 0\}) \cap \Sigma^*$ is regular (cf. e.g. [22]). Hence there exists a regular substitution $\rho : V \to REG$ with $\rho(\alpha) = L(G_\alpha)$ $(\alpha \in V)$ and consequently $L = \rho\tau_0(L_0) \in REG/(K/K_1)$. □

To each extension introduced above we associate an AFL-structure as follows.

Definition. Let X be a symbol from $\{H, H_1, A, R, F, P, S, C, M\}$.

A family K is a full X-AFL if

(i) K is a pre-quasoid, i.e. $\Pi(K) = K$, and

(ii) K is equal to its X-extension, i.e. $X(K) = K$.

In the subsequent two sections we relate full X-AFL's to well-known AFL-structures.

Examples.

1. $H(ONE) = EDTOL$; $H(FIN) = H(REG) = H(ETOL) = ETOL$; $H(INDEX) = INDEX$, hence ETOL and INDEX are full H-AFL's [24, 23, 4].
2. $H_1(ONE) = EDOL$; $H_1(FIN) = EOL$ [24].
3. $A(FIN) = A(REG) = A(CF) = CF$ [25, 26, 14, 15], hence CF is a full A-AFL.
4. $R(FIN) = R(REG) = REG$ (cf. 2.6 and 3.3 below), hence REG is a full R-AFL.
5. For each X from $\{M, C, S, P, F\}$ we have $X(FIN) = X(REG) = REG$ (cf. 2.6) and consequently REG is a full X-AFL.

3. CANONICAL FORMS FOR FULL H-, H_1-, A- AND R-AFL'S.

In [2] it was already shown that a family is a full H-AFL iff it is a full hyper-AFL (cf. section 1, [24, 23]). By a similar argument we obtain that a family is a full H_1-AFL iff it is a full hyper(1)-AFL. It is straightforward to show that the notions of full algebraic AFL (i.e. full A-AFL) and full super-AFL are equivalent (cf. [14]). However, establishing the equivalence between the concepts of full rational AFL (i.e. full R-AFL) and of full substitution-closed AFL is

more complicated. The proof is based on two lemmas which are of some interest on their own.

Lemma 3.1. If K is α-simple, then $K_\infty \subseteq R(K)$.

Proof: Since SYMBOL $\subseteq$ K we have $K \subseteq K/K$ and hence by 2.1.(4) $K_\infty = [K/]^* K$. Moreover $K \subseteq K/K$ also implies $[K/]^{n+1}K = K/(\bigcup_{i=0}^{n} [K/]^i K) = K/[K/]^n K$.

In order to establish the inclusion $K_\infty \subseteq R(K)$ we show that for all natural numbers p, $[K/]^p K \subseteq R(K)$. The proof is by induction on p.

Initial step (p = 0) : $[K/]^0 K = K$. Let $L \subseteq \Sigma^*$ be in K and consider the context-free K-grammar $G = (V,\Sigma,\tau,S)$ where $V = \Sigma \cup \{S\}$, $\tau(S) = L \cup \{S\}$, $\tau(a) = \{a\}$ for each a in Σ. Clearly G is not self-embedding and L(G) = L is in R(K).

Induction step: Suppose $[K/]^p K \subseteq R(K)$, then we have to show that $[K/]^{p+1}K \subseteq R(K)$. By the induction hypothesis we obtain $[K/]^{p+1}K = K/[K/]^p K \subseteq K/R(K)$. Thus it only remains to prove that $K/R(K) \subseteq R(K)$.

Let $L \subseteq \Sigma^*$ be a language in R(K) generated by the regular K-grammar $G = (V,\Sigma,\tau,S)$ and let g be a K-substitution on Σ with $\bigcup\{g(a) \mid a \in \Sigma\} \subseteq \Sigma_0^*$. Without loss of generality we may assume that $V \cap \Sigma_0 = \emptyset$. Consider the context-free K-grammar $H = (V_0,\Sigma_0,U_0,S)$ where $V_0 = V \cup \Sigma_0$, $U_0 = \{\tau_0,\tau_1\}$ with

$\tau_0(\alpha) = \{\alpha\}$ iff $\alpha \in V_0 - \Sigma$

$\tau_0(\alpha) = \{\alpha\} \cup g(\alpha)$ iff $\alpha \in \Sigma$

$\tau_1(\alpha) = \tau(\alpha)$ iff $\alpha \in V$

$\tau_1(\alpha) = \{\alpha\}$ iff $\alpha \in \Sigma_0$

Then clearly H is not self-embedding and $L(H) = g(L(G)) = g(L) \in R(K)$. This completes the induction and establishes the inclusion $K_\infty \subseteq R(K)$. □

Lemma 3.2. If K is a pre-quasoid, then $R(K) \subseteq K_\infty \cup REG$.

Proof: If K is a pre-quasoid but not a quasoid, then K = FIN and hence $R(FIN) = REG \subseteq K_\infty \cup REG$.

Let K be a quasoid, then $REG \subseteq K \subseteq K_\infty$. By 2.5.(2) we only have to prove that $R_1(K) \subseteq K_\infty$. We first show that $R_1(K_\infty) \subseteq K_\infty$. The proof of this inclusion is a generalization of the argument that each not self-embedding context-free grammar generates a regular language (cf.[22]).

Consider a 1-restricted regular K_∞-grammar $G = (V,\Sigma,\tau,S)$. Without loss of generality we may assume that $\tau(a) = \{a\}$ for each a in Σ. (Otherwise we introduce for each a in Σ a new nonterminal symbol A_a and we define an isomorphism ϕ by $\phi(a) = A_a$ $(a \in \Sigma)$, $\phi(\alpha) = \alpha$ $(\alpha \in V - \Sigma)$.We replace τ by τ_0 with $\tau_0(\alpha) = \phi\tau(\alpha)$ iff $\alpha \in V - \Sigma$, $\tau_0(a) = \{a\}$ iff $a \in \Sigma$, $\tau_0(A_a) = \{A_a,a\} \cup \phi\tau(a)$ for each a in Σ).

Moreover we may assume that for each α in V there is a sequence u in $\{\tau\}^+$ such that there is a word in u(S) that contains an occurrence of α. Otherwise we can remove α and intersect all languages involved in G with $(V - \{\alpha\})^*$ (due to the fact that K_∞ is a full AFL (2.4.)) without affecting L(G).

We consider the following cases:

Case 1: For each $A \in V - \Sigma$ there is a sequence u in $\{\tau\}^+$ such that u(A) contains a word in which S occurs.

If $w \in \tau(A)$ is an arbitrary word containing a nonterminal symbol, then it is of one of the four forms: (i) $w = \phi B\psi$, (ii) $w = \phi B$, (iii) $w = B\psi$ or (iv) $w = B$, where ϕ and ψ are nonempty words over V, and $B \in V - \Sigma$. If w satisfies (i) we must have by the assumption of Case 1: there exist sequences u_1 and u_2 in $\{\tau\}^+$ such that:

$u_1u_2\tau(A) \supseteq u_1u_2(\phi B\psi) \supseteq u_1(\phi\phi_1 S\psi_1\psi) \supseteq \{\phi\phi_1\phi_2 A\psi_2\psi_1\psi\}$ for some (possibly empty) words ϕ_1, ϕ_2, ψ_1 and ψ_2 over V. But then, since ϕ and ψ are nonempty, G is a self-embedding context-free K_∞-grammar which contradicts the assumption that G is regular. We obtain the same contradiction if $\tau(A)$ contains words of both forms (ii) and (iii). Thus if $\tau(A)$ contains a word of the form (ii), then in all words of the form (ii) contained in $\tau(A)$ the word ϕ is in Σ^* (Otherwise $\tau(A)$ would contain also a word of the form (i) or (iii)). Hence G is a "right-linear" context-free K_∞-grammar, i.e. for each $A \in V - \Sigma$ we have $\tau(A) \subseteq \Sigma^*(V - \Sigma) \cup \Sigma^*$. By a similar argument we conclude that if $\tau(A)$ contains a word of the form (iii) then G is "left-linear" (i.e. for each $A \in V - \Sigma$: $\tau(A) \subseteq (V - \Sigma)\Sigma^* \cup \Sigma^*$).

So if G is right-linear then we have $\tau(A) = \bigcup\{L_{AX}\{X\} \mid X \in V - \Sigma\} \cup L_A$ for each A in $V - \Sigma$, where L_A and each L_{AX} are languages over Σ. (The left linear case is similar). Note that L_{AX} is empty whenever X does not occur in $\tau(A)$. Since K_∞ is a full AFL (2.4) we have that L_A and each (nonempty) L_{AX} are in K and consequently $R_1(K_\infty) \subseteq K_\infty/REG = K_\infty$ (cf. 2.4 and [9, 11]).

Case 2: There exists a nonterminal symbol A such that for no words ϕ and ψ in V^* and for no sequence U in $\{\tau\}^*$, $\phi S\psi \in u(A)$.

The proof of L(G) being in K_∞ proceeds by induction on the number of nonterminal symbols m. For m = 1, there is nothing to prove because $\lambda(S) = \{S\}$.

Assume that the assertion holds for m = n. Let the number of nonterminal symbols in $V - \Sigma$ be n + 1. Consider the context-free K_∞-grammar $G_1 = (V - \{S\}, \Sigma, \tau_1, A)$

with $\tau_1(\alpha) = \tau(\alpha) \cap (V - \{S\})^*$ iff $\alpha \neq S$

and the context-free K_∞-grammar $G_2 = (V, \Sigma \cup \{A\}, \tau_2, S)$

with $\tau_2(A) = \{A\}$; $\tau_2(\alpha) = \tau(\alpha)$ iff $\alpha \neq A$

Then both G_1 and G_2 are not self-embedding grammars having n nonterminal symbols. Now both of the languages $L(G_1)$ and $L(G_2)$ are in K_∞; either by the induction hypothesis or by Case 1. But L(G) is the result of substituting $L(G_1)$ for A in $L(G_2)$. The fact that K_∞ is substitution-closed implies that L(G) is in K_∞. This completes the induction and establishes the inclusion $R_1(K_\infty) \subseteq K_\infty$.

Finally, together with $K \subseteq K_\infty$ this yields: $R_1(K) \subseteq R_1(K_\infty) \subseteq K_\infty$. □

Theorem 3.3. (1) If K is a pre-quasoid, then $R(K) = K_\infty \cup REG$.

(2) K is a full R-AFL if and only if K is a full substitution-closed AFL.

Proof: (1) Lemmas 3.1 and 3.2.

(2) Let K be a full R-AFL. Then $FIN \subseteq K$ and consequently $REG \subseteq R(K) = K$. Now (1) implies $R(K) = K = K_\infty$. Hence by 2.4 K is a full substitution-closed AFL.

Conversely, let K be a full substitution-closed AFL. Obviously, K is a pre-quasoid and by (1) : $K = K_\infty = R(K)$, i.e. K is a full R-AFL. □

Let $H_1^0(K) = K$, $H_1^{n+1}(K) = H_1 H_1^n(K)$ $n \geq 0$ and $H_1^*(K) = \bigcup_{n=0}^{\infty} H_1^n(K)$.

In the remaining part of this section X will denote a symbol from $\{H, H_1^*, A, R\}$. Let $\hat{\mathcal{X}}(K)$ be the least corresponding full X-AFL containing K (i.e. $\hat{\mathcal{X}}$ equals $\hat{\mathcal{H}}$, $\hat{\mathcal{H}}_1$, $\hat{\mathcal{A}}$ and $\hat{\mathcal{R}}$ respectively).

We now prove the main result of this section, i.e. we will decompose the operator $\hat{\mathcal{X}}$ into simpler operators like X, Π, Δ and Φ. Some other factorizations were already known. E.g. Greibach [14] established a result that may be interpreted as $\hat{\mathcal{A}}(K) = \Delta A\, \hat{\mathcal{H}}(K)$. Similarly from 2.4 due to Ginsburg & Spanier [11] we can infer that $\hat{\mathcal{R}}(K)$ equals the substitution closure of $\Pi(K)$ provided K contains an infinite language.

Theorem 3.4. (Canonical Form Theorem for full H-, H_1-, A- and R-AFL's)

(1) $\hat{\mathfrak{X}}(K) = X\Pi(K)$

(2) If K is nontrivial, then $\hat{\mathfrak{X}}(K) = X\Delta\Phi(K)$.

Proof: (1) We distinguish the following cases:

X = H: In [2] we proved that H(K) is a full H-AFL, provided K is a pre-quasoid. Hence $\hat{\mathfrak{X}}(K) \subseteq H\Pi(K)$. Conversely we have $K \subseteq \hat{\mathfrak{X}}(K)$ which implies $H\Pi(K) \subseteq H\Pi\hat{\mathfrak{X}}(K) = \hat{\mathfrak{X}}(K)$.

$X = H_1^*$: Similar to the previous case it suffices to show that $H_1^*(K)$ is a full H_1-AFL, whenever K is a pre-quasoid. So let K be a pre-quasoid. Obviously $H_1^*(K)$ is H_1-closed and thus it remains to prove that $H_1^*(K)$ is a pre-quasoid. This will be done by induction. Since $FIN \subseteq K$ we have $H_1(EOL) \subseteq H_1^2(K)$. Applying a result from [23] yields that $H_1^2(K)$ is a full AFL. Suppose $H_1^n(K)$, $n \geq 2$ is a full AFL. Then $H_1^{n+1}(K) = H_1H_1^n(K)$ is also full AFL [23], which completes the induction.

X = A: Let K be a pre-quasoid. We will show that A(K) is a full A-AFL. Since the proof that A(K) is a pre-quasoid is rather standard (cf. [26, 2]) and the inclusion $A(K) \subseteq AA(K)$ is trivial, it remains to establish that $AA(K) \subseteq A(K)$.

Let $G = (V,\Sigma,\tau,S)$ be a context-free A(K)-grammar, i.e. τ is a nested A(K)-substitution. For each α in V let $G_\alpha = (V_\alpha,V,\tau_\alpha,S_\alpha)$ be a context-free K-grammar (i.e. each τ_α is a nested K-substitution) such that $L(G_\alpha) = \tau(\alpha)$. Obviously we may assume that all nonterminal alphabets $V_\alpha - V$ are mutually disjoint. Thus we have to show that $L(G) \in A(K)$.

We modify each G_α in such a way that $\tau_\alpha(\beta) = \{\beta\}$ for each β in V (cf. proof of 3.2). Consider the context-free K-grammar $H = (V_0,\Sigma,U_0,S)$ where $V_0 = \bigcup\{V_\alpha \mid \alpha \in V\}$ and $U_0 = \{\sigma_\alpha \mid \alpha \in V\}$ with for each α in V:

$\sigma_\alpha(\beta) = \tau_\alpha(\beta)$ iff $\beta \in V_\alpha - V$

$\sigma_\alpha(\beta) = \{\beta, S_\beta\}$ iff $\beta \in V$

$\sigma_\alpha(\beta) = \{\beta\}$ iff $\beta \in V_0 - V_\alpha$

Clearly $L(H) = L(G)$ and hence $L(G)$ is in $A(K)$.

X = R: The fact that $R(K)$ is a full R-AFL whenever K is a pre-quasoid, directly follows from 3.3 and 2.4.

(2) By (1) and 2.2 we have $\hat{\mathcal{H}}(K) = H\Theta\Delta\Phi(K) \supseteq H\Delta\Phi(K)$. Thus it suffices to show that $H\Theta\Delta\Phi(K) \subseteq H\Delta\Phi(K)$.

Let $L \subseteq \Omega^*$ be a nontrivial language in the family K. Define a finite substitution f by $f(\alpha) = \{\lambda,\omega\}$ for each α in Ω, where ω is an arbitrary but fixed word. Then clearly $\{\omega\} = f(L) \cap \{\omega\}$ is in $\Delta\Phi(K)$, i.e. $ONE \subseteq \Delta\Phi(K)$, which enables us to replace homomorphisms (according to Θ) by $\Delta\Phi(K)$-substitutions as follows. Let $G = (V,\Sigma,U,S)$ be a $\Theta\Delta\Phi(K)$-iteration grammar and consider the $\Delta\Phi(K)$-iteration grammar $H = (V_0,\Sigma,U_0,S)$ where $V_0 = \bigcup\{\phi_{\alpha\tau}(V) \mid \alpha \in V;\ \tau \in U\} \cup V$ with each $\phi_{\alpha\tau}$ is an isomorphism such that all alphabets in this union are mutually disjoint. For each τ in U we define a τ' in U_0 by $\tau'(\alpha) = L$ iff $\alpha \in V$, $L \subseteq (\phi_{\alpha\tau}(V))^*$ is in $\Delta\Phi(K)$ (Note that $\Delta\Phi(K)$ is closed under isomorphism) and $h_{\alpha\tau}(L) = \tau(\alpha)$ where $h_{\alpha\tau}$ is the relative homomorphism according to Θ.

$\tau'(\alpha) = h_{\alpha\tau}(\alpha)$ iff $\alpha \in \phi_{\alpha\tau}(V)$.

$\tau'(\beta) = \{\beta\}$ for each β in $V_0 - V - \phi_{\alpha\tau}(V)$.

By this construction we have $L(H) = L(G)$ and consequently $\hat{\mathcal{H}}(K) = H\Delta\Phi(K)$ for each nontrivial K.

Since this construction preserves the number of substitutions and the not self-embedding property, whereas nesting can also easily be incorporated, the same conclusion holds in the other cases. □

As each pre-quasoid contains FIN, and FIN is the smallest pre-quasoid we have

Corollary 3.5. Let X be a symbol from $\{H, H_1^*, A, R\}$. Then

(1) each full X-AFL contains the family X(FIN)

(2) X(FIN) is the smallest full X-AFL.

This corollary implies that H(FIN) = ETOL is the smallest full H-AFL [4], A(FIN) = CF is the smallest full A-AFL [14] and R(FIN) = REG is the smallest full R-AFL [11].

An improvement in the H_1-case to $\hat{\mathfrak{H}}_1(K) = H_1\Pi(K)$ instead of $H_1^*\Pi(K)$ is impossible because $H_1(FIN) = EOL$ is not even a full AFL [16]. Starting with quasoids rather than pre-quasoids indeed yields full AFL's [23] but no full H_1-AFL or even full R-AFL's [4]. Recently Engelfriet [8] showed that the iteration of H_1 (applied to $\Pi(K)$) cannot be reduced to a finite power since $H_1^*(FIN)$ (i.e. the smallest full H_1-AFL) gives rise to an infinite hierarchy of full AFL's:
$H_1^2(FIN) \subset H_1^3(FIN) \subset \ldots \subset H_1^n(FIN) \subset \ldots$

For X equal to H, H_1, A and R we denote the class of all full X-AFL's (over Σ_ω) by X-AFL.

Corollary 3.6. R-AFL $\supset$ A-AFL $\supset$ H_1-AFL $\supset$ H-AFL.

Proof: From the definitions in 2 it is clear that R-AFL $\supseteq$ A-AFL $\supseteq$ H_1-AFL $\supseteq$ H-AFL. Since the smallest full X-AFL's for $X \in \{R, A, H_1, H\}$ are mutually different (i.e. REG $\subset$ CF $\subset$ $H_1^*(FIN)$ $\subset$ ETOL; cf. 3.5 and [21, 8]) the inclusions are also proper. □

4. CANONICAL FORMS FOR SUBRATIONALLY CLOSED FULL AFL's.

Let X be a symbol from {M,C,S,P,F}. First we relate full X-AFL's

to well-known AFL-structures.

In this section we only give detailed proofs for a few typical cases. The other arguments are obtained by straightforward modifications and are left as simple exercises.

Theorem 4.1.

(1) K is a full M-AFL iff K is a full trio

(2) K is a full C-AFL iff K is a full Kleene-AFL

(3) K is a full S-AFL iff K is a full semi-AFL

(4) K is a full P-AFL iff K is a full pseudo-AFL

(5) K is a full F-AFL iff K is a full AFL.

Proof: (5) Let K be a full F-AFL, i.e. K is a pre-quasoid and $K = \underline{R}(REG,K,FIN)$. By 2.6 we have $K = REG/K/REG$. This implies (i) K is closed under Δ, (ii) $REG \subseteq K$, because $SYMBOL \subseteq K$, (iii) $REG/K \subseteq K$ and $K/REG \subseteq K$, since $SYMBOL \subseteq REG$, which means that K is a full AFL [11].

Conversely, let K be a full AFL. Clearly K is a pre-quasoid. Moreover $K/REG \subseteq K$ and $REG/K \subseteq K$ hold [11]. Applying 2.6 yields $R(REG,K,FIN) = REG/K/REG \subseteq K$, whereas the opposite inclusion is obvious. □

Let $\hat{\mathfrak{X}}(K)$ denote the least full X-AFL containing K, i.e. if X equals M, C, S, P or F then $\hat{\mathfrak{X}}$ is equal to $\hat{\mathfrak{M}}$, $\hat{\mathfrak{C}}$, $\hat{\mathfrak{S}}$, $\hat{\mathfrak{P}}$ or $\hat{\mathfrak{F}}$ respectively. With respect to Canonical Forms the case X = C differs from the other cases and will be treated separately (cf. 4.4).

Theorem 4.2. (First Canonical Form Theorem for Full M-, S-, P- and F-AFL's).
Let X be a symbol from {M,S,P,F}. Then

(1) $\hat{\mathfrak{X}}(K) = X\Pi(K)$

(2) if K is nontrivial, then $\hat{\mathfrak{X}}(K) = X\Delta\Phi(K)$.

Proof: (1) X = S. Similar as in the proof of 3.4 it suffices to show that S(K) is a full S-AFL provided K is a pre-quasoid. So let K be a pre-quasoid, then by 2.6 S(K) = REG/K/ALPHA and hence

(i) $\Phi S(K) = FIN/REG/K/ALPHA = REG/K/ALPHA = S(K)$.

(ii) $\Delta S(K) = \Delta(REG/K/ALPHA) \subseteq \Delta(REG)/\Delta\Phi(K/ALPHA) =$
$REG/\Delta(K/ALPHA) = REG/\{(\bigcup_i L_i) \cap R \mid L_i \in K;\ R \in REG\} =$
$REG/\{\bigcup_i (L_i \cap R) \mid L_i \in K;\ R \in REG\} = REG/K/ALPHA = S(K)$
(The inclusion is obtained by 2.3.(1); i goes through any finite index set).

(iii) $SS(K) = REG/REG/K/ALPHA/ALPHA = REG/K/ALPHA = S(K)$.

(i) - (iii) imply that S(K) is a full S-AFL.

X = F: Apart from taking REG instead of ALPHA, the cases (i) and (iii) are the same as above. So it remains to show that $\Delta F(K) \subseteq F(K)$. Applying 2.3.(1) twice yields: $\Delta F(K) = \Delta(REG/K/REG) \subseteq \Delta(REG)/\Delta\Phi(K/REG) =$ $REG/\Delta(K/REG) \subseteq REG/\Delta(K)/\Delta\Phi(REG) = REG/K/REG = F(K)$.

(2) Let K_X be SYMBOL, ALPHA, FIN and REG for X equal to M, S, P and F respectively. By (1), 2.2 and the fact that $\Delta\Phi(K)$ is closed under isomorphism we obtain $\hat{\mathfrak{X}}(K) = X\Pi(K) = X\Theta\Delta\Phi(K) = REG/ONE/\Delta\Phi(K)/K_X =$ $REG/\Delta\Phi(K)/K_X = X\Delta\Phi(K)$. □

From 2.6 and 4.2.(1) we may infer other Canonical Forms like $\hat{\mathfrak{F}}(K) = \hat{\mathfrak{M}}(K)/REG$ or $\hat{\mathfrak{Y}}(K) = \hat{\mathfrak{M}}(K)/ALPHA$ which were originally established in [10] and [19] respectively (cf. [12, 20]).

Theorem 4.3. (Second Canonical Form Theorem for Full M-, S-, P- and F-AFL's). If K is σ-simple, then $\hat{\mathfrak{X}}(K) = \Pi X(K) = \Theta\Delta X(K)$ for each X from {M,S,P.F}.

Proof: The inclusion $K \subseteq \Pi(K)$ and 4.2 imply $X(K) \subseteq X\Pi(K) = \hat{\mathcal{X}}(K)$ and hence $\Theta\Delta X(K) \subseteq \Pi X(K) \subseteq \Pi\, \hat{\mathcal{X}}(K) = \hat{\mathcal{X}}(K)$. Thus it remains to show that $\hat{\mathcal{X}}(K) \subseteq \Theta\Delta X(K)$. Distinguish the following cases.

X = M: According to 4.2, 2.6 and 2.3(2) we have $\hat{\mathcal{M}}(K) = REG/\Delta\Phi(K)$ $\subseteq \Theta\Delta((REG \cup FIN)/\Phi\Phi(K)) = \Theta\Delta M(K)$.

X = S: By 4.2 and the previous case we have: $\hat{\mathcal{S}}(K) = \{\bigcup L_i \mid L_i \in \hat{\mathcal{M}}(K);$ $1 \leq i \leq n, n \geq 1\} = \{\bigcup L_i \mid L_i \in \Theta\Delta M(K);\ 1 \leq i \leq n, n \geq 1\}$, i.e. $L_i = h_i(L_i' \cap R_i)$ with $L_i' \in M(K)$, $R_i \in REG$ and h_i is an arbitrary homomorphism. We may assume that $L_i' \subseteq \Sigma_i^*$ where all the Σ_i are mutually disjoint alphabets. Let $R = \bigcup(R_i \cap \Sigma_i^*)$, $1 \leq i \leq n$ and let h be the homomorphism defined by $h(\alpha) = h_i(\alpha)$ for each α in $\bigcup \Sigma_i$. It is straightforward to show that $\bigcup h_i(L_i' \cap R_i) = h((\bigcup L_i) \cap R)$, $1 \leq i \leq n$, which implies that $\hat{\mathcal{S}}(K) \subseteq \Theta\Delta\{\bigcup L_i' \mid L_i' \in M(K);\ 1 \leq i \leq n,$ $n \geq 1\} = \Theta\Delta S(K)$.

X = P: (and similar for X = F): From 4.2 and 2.6 we obtain $\hat{\mathcal{P}}(K) = REG/\Delta\Phi(K)/FIN$. Applying 2.3.(2) twice, yields: $\hat{\mathcal{P}}(K) \subseteq$ $REG/\Theta\Delta((\Phi(K) \cup FIN)/\Phi(FIN)) = REG/\Delta(\Phi(K)/FIN) \subseteq \Theta\Delta((REG \cup FIN)/\Phi(\Phi(K)/FIN)) =$ $\Theta\Delta(REG/K/FIN) = \Theta\Delta P(K)$. □

Other Canonical Forms like $\hat{\mathcal{F}}(K) = \hat{\mathcal{M}}(K/REG)$ and $\hat{\mathcal{G}}(K) = \hat{\mathcal{M}}(K/ALPHA)$ (cf. [19, 20]) for σ-simple K, directly follow from 4.3.

In the case X = C it is possible to establish a Canonical Form like $\hat{\mathcal{C}}(K) = \Pi C(K) = \Theta\Delta C(K)$ (cf. 4.3) provided K is σ-simple and $L\alpha$ is in K for each L in K and each symbol α not occurring in any word of L. In order to avoid further restrictions on K we will however follow another approach.

A substitution $\tau : V \to K$ is called marked if $\tau(\alpha) \subseteq \Delta^*\alpha$ for each α in V, and the alphabets Δ and V are disjoint. Let $K_1 \not\subseteq K_2$ denote the family obtained by applying marked K_1-substitutions on languages from

K_2, i.e. $K_1 \swarrow K_2 = \{\tau(L) \mid L \in K_2, \tau \text{ is a marked } K_1\text{-substitution}\}$ (cf. [19]). We now redefine the C-extension by $C(K) = REG/(K \swarrow STAR)$ (cf. 2.6. Notice that 4.1(2) remains valid). In the sequel we only consider this redefined extension, for which we prove

Theorem 4.4. (Canonical Form Theorem for Full C-AFL's).
$\hat{\mathcal{C}}(K) = \Pi C(K) = \Theta\Delta C(K)$.

Proof: Since $\hat{\mathcal{C}}(K)$ is closed under Π and C we have $\Pi C(K) \subseteq \hat{\mathcal{C}}(K)$. In order to establish the converse inclusion it suffices to show that $\Pi C(K)$ is a full C-AFL containing K. Obviously $\Pi C(K)$ is a pre-quasoid and it is easy to prove that $K \subseteq \Pi C(K)$. Thus it remains to show that $\Pi C(K)$ is closed under C. The proof is based on 2.2, 2.3.(2) and the following inclusions (i) $\Delta(K) \swarrow STAR \subseteq \Theta\Delta(K \swarrow STAR)$, (ii) $(K \swarrow STAR) \swarrow STAR \subseteq \Delta\Phi(K \swarrow STAR)$.

Let $L \subseteq \Sigma^*$ be in K, $R \in REG$ and let a, b, $c \notin \Sigma$. Define a homomorphism h and a finite substitution f by $h(b) = a$, $h(\alpha) = \alpha$ for $\alpha \in \Sigma$, and $f(c) = \{a,ab\}$, $f(\alpha) = \{\alpha\}$ for $\alpha \in \Sigma$. It is straightforward to prove that $((L \cap R)a)^* = h((Lb)^* \cap (Rb)^*)$ and $((La)^*b)^* = f((Lc)^*) \cap (\Sigma \cup \{a,b\})^*b$, which establishes (i) and (ii) respectively.

Applying respectively 2.2, (i), 2.3(2), (ii) and 2.3(2) yields

$C\Pi C(K) = REG/\Delta\Phi C(K) \swarrow STAR \subseteq REG/\Theta\Delta(\Phi C(K) \swarrow STAR)$

$\subseteq \Theta\Delta(REG/\Phi C(K) \swarrow STAR) = \Pi(REG/(K \swarrow STAR) \swarrow STAR)$

$\subseteq \Pi(REG/\Delta\Phi(K \swarrow STAR)) \subseteq \Pi(REG/\Phi(K \swarrow STAR)) = \Pi C(K)$,

i.e. $\Pi C(K)$ is C-closed and hence $\hat{\mathcal{C}}(K) = \Pi C(K)$. Finally we have $\Pi C(K) = \Theta\Delta(FIN/REG/(K \swarrow STAR)) = \Theta\Delta C(K)$. □

From 4.2 - 4.4 we obtain

Corollary 4.5. Let X be a symbol from {M,S,P,C,F}. Then

(1) each full X-AFL contains the family X(FIN).

(2) X(FIN) = REG is the smallest full X-AFL.

By X-AFL we will again denote the class of all full X-AFL's over Σ_ω (for X = M,C,S,P of F).

Theorem 4.6.

(1) M-AFL $\supset$ S-AFL $\supset$ P-AFL $\supset$ F-AFL $\supset$ R-AFL

(2) M-AFL $\supset$ C-AFL $\supset$ F-AFL

(3) C-AFL is incomparable with S-AFL and P-AFL.

Proof: We will establish the existence of a full C-AFL K_0 which is not closed under union. Together with well-known results (cf. [9, 10]) this implies 4.6. (1) - (3).

Let K_1 and K_2 be incomparable full AFL's (The existence of K_1 and K_2 is guaranteed by [13]). Let $L_1 \subseteq \Sigma_1^*$ be in $K_1 - K_2$ and let $L_2 \subseteq \Sigma_2^*$ be in $K_2 - K_1$ such that $\Sigma_1 \cap \Sigma_2 = \emptyset$. For i = 1, 2 define homomorphisms h_i on $\Sigma_1 \cup \Sigma_2$ by $h_i(\alpha) = \alpha$ iff $\alpha \in \Sigma_i$ and $h_i(\alpha) = \lambda$ iff $\alpha \notin \Sigma_i$.

We define $K_0 = K_1 \cup K_2$, which is a full C-AFL [3]. Suppose K_0 is closed under union. Then according to [9, 10] K_0 is also closed under concatenation and consequently $L_1L_2 \in K_1$ or $L_1L_2 \in K_2$. But then we have $h_2(L_1L_2) = L_2 \in K_1$ or $h_1(L_1L_2) = L_1 \in K_2$ respectively, contradicting the choice of L_1 and L_2. □

5. SUBSTITUTING FAMILIES INTO FAMILIES.

In this section we apply Canonical Forms of sections 3 and 4 to the family obtained by substituting K_1-languages into languages from K_2 (5.1 - 5.3). Then we discuss a generalization of 4.2 - 4.4 and 5.2 (5.4 - 5.5) and finally, we consider an example of the application of Canonical Forms and related results in proving a certain family to be a full X-AFL (5.7.).

We first consider substituting a pre-quasoid into a pre-quasoid.

Lemma 5.1. If K_1 is σ-simple and if K_2 is nontrivial, then $\Pi(K_1/K_2) \subseteq \Pi(K_1)/\Delta\Phi(K_2) = \Pi(K_1)/\Pi(K_2) = \Pi(K_1/FIN/K_2)$, whereas equality holds whenever, either K_1 is closed under substitution into FIN (i.e. under union and concatenation), or K_2 is closed under finite substitution.

Proof: An application of 2.2. and 2.3(1) yields $\Pi(K_1/K_2) = \Theta\Delta(\Phi(K_1)/K_2) \subseteq ONE/\Delta\Phi(K_1)/\Delta\Phi(K_2) = \Pi(K_1)/\Delta\Phi(K_2) \subseteq \Pi(K_1)/\Pi(K_2)$. By 2.2 and 2.3(2) we have $ONE/\Delta\Phi(K_1)/ONE/\Delta\Phi(K_2) \subseteq \Theta\Delta(\Phi(K_1)/\Phi(ONE))/\Delta\Phi(K_2) = \Theta\Delta(\Phi(K_1)/FIN)/\Delta\Phi(K_2) \subseteq \Theta\Delta(\Phi(K_1))/FIN/\Phi(K_2) = \Pi(K_1/FIN/K_2)$.

Using the former inclusion we just established with FIN/K_2 instead of K_2 we obtain $\Pi(K_1/FIN/K_2) \subseteq \Pi(K_1)/\Delta\Phi(K_2)$. □

From 5.1. it directly follows that substituting a pre-quasoid into a pre-quasoid yields a pre-quasoid. A similar well-known conclusion for full trio, semi-AFL, AFL [9, 11] and also for full Kleene-AFL and pseudo-AFL can be immediately inferred from

Theorem 5.2. Let K_1 and K_2 be σ-simple families. If K_2 is also closed under finite substitution, then

(1) $\hat{\mathcal{C}}(K_1/K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2 \mathbin{/\!\!/} \text{STAR})$

(2) $\hat{\mathcal{AL}}(K_1/K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2)$

(3) $\hat{\mathcal{F}}(K_1/K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2)/\text{ALPHA}$

(4) $\hat{\mathcal{F}}(K_1/K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2)/\text{FIN}$

(5) $\hat{\mathcal{F}}(K_1/K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2)/\text{REG}$

Proof: (1) By 4.4 and 5.1 we have $\hat{\mathcal{C}}(K_1/K_2) = \Pi(\text{REG}/K_1/K_2 \mathbin{/\!\!/} \text{STAR}) = \Pi(\text{REG}/K_1)/\Delta\Phi(K_2 \mathbin{/\!\!/} \text{STAR})$, and hence by 4.3 $\hat{\mathcal{C}}(K_1/K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2 \mathbin{/\!\!/} \text{STAR})$.
(2) According to 4.2 and 5.1 we obtain $\hat{\mathcal{AL}}(K_1/K_2) = \text{REG}/\Pi(K_1/K_2) = \text{REG}/\Pi(K_1)/\Delta\Phi(K_2) = \hat{\mathcal{AL}}(K_1)/\Delta(K_2)$.

Together with 2.6 and 4.2 this implies (3) - (5). □

In the remaining cases we even obtain

Theorem 5.3. Let X be a symbol from $\{R,A,H_1^*,H\}$. Then

(1) $\hat{\mathcal{X}}(K_1 \cup K_2) = X(\Pi(K_1) \cup \Pi(K_2)) = X(\Pi(K_1)/\Pi(K_2)) = X(\Pi(K_2)/\Pi(K_1))$

(2) $\hat{\mathcal{X}}(K_1/K_2) = \hat{\mathcal{X}}(K_2/K_1) = \hat{\mathcal{X}}(K_1 \cup K_2)$ provided both K_1 and K_2 are σ-simple.

Proof: (1) For reasons of symmetry it suffices to show the former two equalities. By 3.4(1) and [3] we have $\hat{\mathcal{X}}(K_1 \cup K_2) = X\Pi(K_1 \cup K_2) = X(\Pi(K_1) \cup \Pi(K_2))$. From 3.6 and 3.3(2) we obtain that each full X-AFL is a substitution-closed pre-quasoid (for $X \in \{R,A,H_1^*,H\}$), which implies $\Pi(K_1)/\Pi(K_2) \subseteq \hat{\mathcal{X}}(K_1 \cup K_2)$. Since each pre-quasoid includes SYMBOL, we also have $K_1 \cup K_2 \subseteq \Pi(K_1)/\Pi(K_2) \subseteq \hat{\mathcal{X}}(K_1 \cup K_2)$. Applying 3.4(1) and 5.1 (with $\Pi(K_i)$ instead of K_i, $i = 1,2$) yields $\hat{\mathcal{X}}(K_1 \cup K_2) = X\Pi(\Pi(K_1)/\Pi(K_2)) \subseteq X(\Pi(K_1)/\Pi(K_2)) \subseteq \hat{\mathcal{X}}(K_1 \cup K_2)$.
(2) If both K_1 and K_2 are σ-simple, then $K_1 \cup K_2 \subseteq K_1/K_2 \subseteq \hat{\mathcal{X}}(K_1 \cup K_2)$. Hence $\hat{\mathcal{X}}(K_1 \cup K_2) = \hat{\mathcal{X}}(K_1/K_2)$. □

We may also consider 4.2 - 4.4 and 5.2 as particular instances or more general results (cf. 5.4 and 5.5 below). We call a family K a full [K',K"]-structure if K is a pre-quasoid which is closed under K'-substitution and under substitution into K", i.e. $K'/K/K'' \subseteq K$ (cf. [19]). If K" = STAR we demand, as in 4, that K is closed under marked substitution into STAR, i.e. $K'/K/STAR \subseteq K$. Let [K',K"](K) denote the smallest full [K',K"]-structure containing K.

Theorem 5.4. Let K' be a pre-quasoid and let K" be either a pre-quasoid or equal to SYMBOL or ALPHA. Then

(1) $[K',STAR](K) = \Pi(K'_\infty/(K/STAR)) = \Theta\Delta(K'_\infty/(K/STAR))$

(2) $[K',K''](K) = K'_\infty/\Pi(K)/K''_\infty = K'_\infty/\Delta\Phi(K)/K''_\infty = \Pi(K'_\infty/K/K''_\infty) = \Theta\Delta(K'_\infty/K/K''_\infty)$

provided K is σ-simple.

The proof of 5.4. consists of a straightforward modification of the arguments used in establishing 4.2 - 4.4 and it will therefore be omitted. Note that SYMBOL and ALPHA are substitution-closed, i.e. $K''_\infty = K''$.

From 5.1 and 5.4 we can infer in a way similar to 5.2

Theorem 5.5. Let K' be a pre-quasoid and let K" be either a pre-quasoid or equal to SYMBOL or ALPHA. If K_1 and K_2 are σ-simple families and, if K_2 is closed under finite substitution, then

(1) $[K',STAR](K_1/K_2) = \Pi(K'_\infty/K_1/K_2/STAR) = \Theta\Delta(K'_\infty/K_1/K_2/STAR)$.

(2) $[K',K''](K_1/K_2) = K'_\infty/\Pi(K_1/K_2)/K''_\infty = K'_\infty/\Pi(K_1)/\Delta(K_2)/K''_\infty$.

Taking K' (and K") equal to a quasoid instead of a pre-quasoid implies that K'_∞ (and K''_∞) is a full R-AFL (cf. 2.4) and that 5.4 and 5.5 yield full (pseudo-)AFL. In particular we have $\hat{\mathfrak{P}}(K) = [REG,FIN](K)$ and $\hat{\mathfrak{F}}(K) = [REG,REG](K)$, whereas full M-AFL, S-AFL and C-AFL correspond

to the cases K' = REG, K" = SYMBOL, ALPHA and STAR respectively (provided we apply a marked substitution in the last case).

When we take K' = FIN instead of REG we can obtain (1) pre-quasoids (K" = SYMBOL), (2) pre-quasoids closed under union (K" = ALPHA), (3) pre-quasoids closed under concatenation [and union] (K" = FIN) (4) quasoids closed under Kleene * (K" = STAR), and (5) quasoids closed under Kleene *, union and/or concatenation (K" = REG). It is straightforward to show that results similar to 4.2 - 4.6 and 5.2 also hold for these structures.

Canonical Forms and theorems like 5.2 - 5.5 could serve as a useful tool in showing certain families to be a (particular kind of) full AFL or a full [K',K"]-structure. We conclude this paper with an application.

A language L is called hyper-sentential [27] over a family K, if there exist (1) an alphabet V, (2) an initial language $L_0 \subseteq V^*$ in K, (3) a finite set U of K-substitutions over V, such that $L = U^*(L_0)$. The hyper-sentential extension $(K) of K consists of all languages hyper-sentential over K. Clearly $L \cap \Sigma^*$ is in H(K) when L is in $(K). The hyper-algebraic extension H(K) has been characterized in terms of $(K) by Van Leeuwen & Wood [27].

Theorem 5.6. Let K be a family closed under isomorphism and under intersection with Σ^* for each finite alphabet Σ. If ONE $\subseteq$ K, then $H(K) = (SYMBOL \cup \{\lambda\})/\$(K)$.

In the finite case even a stronger result holds: ETOL = SYMBOL/TOL and EOL = SYMBOL/OL (cf. [5,6], note that $(FIN) = TOL). A SYMBOL-[or SYMBOL $\cup$ $\{\lambda\}$-] substitution is sometimes referred to as a [weak] coding.

We show how to obtain full AFL's from $(K).

Theorem 5.7. Let K be a full trio. If K_0 is a pre-quasoid, then $K/\$(K_0)$ is a $[REG, \hat{\mathcal{H}}(K_0)]$-structure and consequently, it is a full AFL.

Proof: $[REG, \hat{\mathcal{H}}(K_0)](K/\$(K_0)) =$ (K is a full trio)

$[REG, \hat{\mathcal{H}}(K_0)](K/(SYMBOL \cup \{\lambda\})/\$(K_0)) =$ (5.6)

$[REG, \hat{\mathcal{H}}(K_0)](K/H(K_0)) =$ (5.5.(2), 3.4)

$REG/K/H(K_0)/H(K_0) =$ (K is a full trio, 3.4)

$K/H(K_0) =$ (5.6)

$K/(SYMBOL \cup \{\lambda\})/\$(K_0) =$ (K is a full trio)

$K/\$(K_0)$, i.e. $K/\$(K_0)$ is a $[REG, \hat{\mathcal{H}}(K_0)]$-structure. □

If K is contained in $H(K_0)$, then clearly $K/\$(K_0) = H(K_0)$ and consequently it is a full H-AFL. Thus 5.7 is only interesting if K is not contained in $H(K_0)$. When $K_0 = FIN$, 5.7 yields: K/TOL is a full AFL provided K is a full trio, which was originally established by Salomaa [23, 4]. Using EOL = SYMBOL/OL [5] one can prove in a way similar to 5.7 that K/OL is a full AFL whenever K is a full trio [23, 4].

References

1. A.V. Aho, Indexed Grammars - An Extension of Context-Free Grammars, J. Ass. Comp. Mach. 15 (1968) 647 - 671.

2. P.R.J. Asveld, Controlled Iteration Grammars and Full Hyper-AFL's, Information and Control 34 (1977) 248 - 269.

3. P.R.J. Asveld, Incomparable Elements in Algebraic Lattices with an Application to AFL-theory, TW-memorandum No. 202, Twente University of Technology, Enschede, The Netherlands.

4. P.A. Christensen, Hyper-AFL's and ETOL Systems, DAIMI PB-35, University of Aarhus, Denmark (Cf. pp. 254 -257 in G.Rozenberg & A. Salomaa (Eds.), "L Systems" (1974) Springer, Berlin-Heidelberg-New York).

5. A. Ehrenfeucht & G. Rozenberg, The Equality of EOL Languages and Codings of OL Languages, Internat. J. Comp. Math. 4 (1974) 95 - 104.

6. A. Ehrenfeucht & G. Rozenberg, Nonterminals Versus Homomorphisms in Defining Languages for Some Classes of Rewriting Systems, Acta Informatica 3 (1974) 265 - 283.

7. C.C. Elgot & J.E. Mezei, On Relations Defined by Generalized Finite Automata, I.B.M. J. Res. Develop. 9 (1965) 47 - 65.

8. J. Engelfriet, Iterating Iterated Substitution, Theor. Comp. Sci. 5 (1977) 85 - 100.

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

10. S. Ginsburg, S.A. Greibach & J.E. Hopcroft, "Studies in Abstract Families of Languages" (1969) Memoirs Amer. Math. Soc. No. 87.

11. S. Ginsburg & E.H. Spanier, Substitution in Families of Languages, Information Sci. 2 (1970) 83 - 110.

12. S. Ginsburg & E.H. Spanier, AFL with the Semilinear Property, J. Comp. System Sci. 5 (1971) 365 - 396.

13. S. Ginsburg & E.H. Spanier, On Incomparable Abstract Family of Languages (AFL), J. Comp. System Sci. 9 (1974) 88 - 108.

14. S.A. Greibach, Full AFL's and Nested Iterated Substitution, Information and Control 16 (1970) 7 - 35.

15. S.A. Greibach, A Generalization of Parikh's Semilinear Theorem, Discrete Math. 2 (1972) 347 - 355.

16. G.T. Herman, Closure Properties of Some Families of Languages Associated with Biological Systems, Information and Control 24 (1974) 101 - 121.

17. G.T. Herman & G. Rozenberg, "Developmental Systems and Languages" (1975) North-Holland, Amsterdam.

18. J.E. Hopcroft & J.D. Ullman, "Formal Languages and Their Relation to Automata" (1969) Addison-Wesley, Reading, Mass.

19. D.J. Lewis, Closure of Families of Languages under Substitution Operators, pp. 100 - 108 in "Second Annual A.C.M. Symposium on Theory of Computing" (1970).

20. M. Nivat, Opérateurs sur les familles de langages, Rapport de Recherche No. 106 (1975) I.R.I.A., Rocquencourt, France.

21. G. Rozenberg, Extension of Tabled OL-Systems and Languages, Internat. J. Comp. Inform. Sci. 2 (1973) 311 - 336.

22. A. Salomaa, "Formal Languages" (1973) Academic Press, New York.

23. A. Salomaa, Macros, Iterated Substitution and Lindenmayer-AFL's, DAIMI PB-18, University of Aarhus, Denmark (cf. pp. 250 - 253 in A. Salomaa & G. Rozenberg (Eds.), "L Systems" (1974) Springer, Berlin-Heidelberg-New York).

24. J. van Leeuwen, F-iteration Languages, Memorandum (1973), University of California, Berkeley.

25. J. van Leeuwen, A Generalization of Parikh's Theorem in Formal Language Theory, pp. 17 - 26 in J. Loeckx (Ed.), "Automata, Languages and Programming, 2nd Colloquium" (1974) Springer, Berlin-Heidelberg-New York.

26. J. van Leeuwen, Effective Constructions in Well-Partially-Ordered Free Monoids, Discrete Math. (to appear).

27. J. van Leeuwen & D. Wood, A Decomposition Theorem for Hyper-Algebraic Extensions of Language Families, Theor. Comp. Sci. 1 (1976) 199 - 214.

Chapter 6

Operators on Language Families and Their Monoids

Reprinted from *TW-memorandum No. 211 (April 1978)*

Abstract

The structure of partially ordered monoids generated by certain operators on language families (of which each is induced by operations on languages) is investigated. In particular operators are considered that define prequasoids (i.e. language families closed under finite substitution and intersection with regular languages), full trios, full semi-AFL's, full pseudo-AFL's (i.e. full semi-AFL's closed under concatenation), full AFL's, full substitution-closed AFL's, full super-AFL's, and full hyper-AFL's. The structure of these monoids provides better insight in the (in)dependency of closure properties relevant in AFL-theory.

1. Introduction

The notion of full abstract family of languages (or full AFL) has been introduced in 1969 by Ginsburg and Greibach [13] in order to investigate in a more general way language families closed under the regular operations (union, concatenation and Kleene *), homomorphism, inverse homomorphism and intersection with regular languages. Nowadays AFL-theory is a well-established part of formal language theory and it constitutes a very fruitful approach in studying algebraic properties of language families. The reader is referred to Ginsburg's book [12] as a standard work covering the main results obtained in this field.

Since a full AFL may be considered as an algebra - i.e. a set A_0 closed under a fixed collection Ω_0 of operations on A_0 - it should be no surprise that certain results in AFL-theory have been inspired (more or less explicitely) by methods developed in universal algebra. In order to mention an example, consider an algebra (A_0,Ω_0) and a collection $\{\Omega_1,\ldots,\Omega_n\}$ $(n \geq 2)$ of proper subsets of Ω_0, which cover Ω_0, i.e. Ω_0 equals the union of $\Omega_1,\ldots,\Omega_n$. The question now is, whether the collection $\{\Omega_1,\ldots,\Omega_n\}$ is independent or not - or, equivalently - does closure under the operations in $\Omega_1,\ldots,\Omega_{k-1},\Omega_{k+1},\ldots,\Omega_n$ $(1 \leq k \leq n)$ imply closure under the operations from Ω_k? With respect to the operations defining a full AFL this problem was first studied by Greibach and Hopcroft [17]. We will investigate this question by studying the structure of the monoid generated by the closure operators corresponding to $\Omega_1,\ldots,\Omega_n$ as follows.

Let A be an arbitrary subset of A_0 and let for each k $(0 \leq k \leq n)$ $\Gamma_k(A)$ denote the smallest set including A which is closed under the operations from Ω_k. In exactly the same way we define for each subset J of $\{1,\ldots,n\}$ $\Gamma_J(A)$ as the smallest set including A which is closed under the operations in $\bigcup_{k\in J} \Omega_k$ (We identify Γ_k with Γ_J whenever J is the singleton $\{k\}$. Note that

$\Gamma_0(A) = \Gamma_{\{1,\ldots,n\}}(A)$ for each $A \subseteq A_0$. For reasons of consistency we define $\Gamma_\emptyset = I$, i.e. the identity operator). In universal algebra the operators $\Gamma_J : P(A_0) \to P(A_0)$, where $P(A_0)$ is the power set of A_0, are usually called closure operators. Since the composition of (closure) operators is associative the collection $\{\Gamma_J \mid J \subseteq \{1,\ldots,n\}\}$ generates a monoid [25]. This monoid is provided with a partial order induced by the set-theoretical inclusion in $P(A_0)$, viz. $\Gamma_J \leq \Gamma_{J'}$ iff for each $A \subseteq A_0$ $\Gamma_J(A) \subseteq \Gamma_{J'}(A)$ [25].

It can be shown that if $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ - i.e. the monoid generated by $\{\Gamma_1,\ldots,\Gamma_n\}$ - is finite, then $Mn\{\Gamma_J \mid J \subseteq \{1,\ldots,n\}\}$ coincides with $Mn\{\Gamma_1,\ldots,\Gamma_n\}$. In all cases to be considered in the sequel $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ turns out to be finite, and therefore we restrict our attention to $Mn\{\Gamma_1,\ldots,\Gamma_n\}$.

The problem whether Ω_k depends on $\{\Omega_1,\ldots,\Omega_{k-1},\Omega_{k+1},\ldots,\Omega_n\}$ can now be rephrased as: is $\Gamma_k \leq \Gamma_J$ for $J = \{1,\ldots,k-1,k+1,\ldots,n\}$? So the partial order of the monoid generated by $\{\Gamma_J \mid J \subseteq \{1,\ldots,n\}\}$ or $\{\Gamma_1,\ldots,\Gamma_n\}$ whenever this monoid is finite, enables us to solve all dependency problems for the collection $\{\Omega_1,\ldots,\Omega_n\}$.

The structure of $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ - i.e. its partial order and its multiplication table - does not only describe all dependency relations in the collection $\{\Omega_1,\ldots,\Omega_n\}$ but it also solves other questions like: is the result of applying $\Gamma'_1,\ldots,\Gamma'_k$ (in that order) in general (in)comparable with the result obtained by application of $\Gamma''_1,\ldots,\Gamma''_m$ (in that order; $\Gamma'_i,\Gamma''_i \in \{\Gamma_1,\ldots,\Gamma_n\}$)? So studying the structure of $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ yields a complete picture of the power of successively applying operators from $\{\Gamma_1,\ldots,\Gamma_n\}$.

The structure of $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ is mainly studied in algebra for the case that A_0 is a variety or primitive class, i.e. a class of algebras closed under taking subalgebras (Γ_S), direct products (Γ_P), and homomorphic images

(Γ_H) [26, 9, 16, 24]. In general $Mn\{\Gamma_S,\Gamma_P,\Gamma_H\}$ is finite [26], and for some special varieties it has been investigated to what extent the structure of $Mn\{\Gamma_S,\Gamma_P,\Gamma_H\}$ (as established in [26] for the general case) collapses [9, 16].

In this paper we study $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ for the cases that Ω_0 corresponds to one of the full AFL-structures surveyed in [2] and $\Omega_1,\ldots,\Omega_n$ is a natural division of Ω_0. Also here we obtain in (almost) all cases to be considered that $Mn\{\Gamma_1,\ldots,\Gamma_n\}$ is finite and for a few monoids we determine their structure completely. This yields a more complete insight in dependency of full AFL-operations (cf. [17]) and in the relative capacity of the successive application of closure operators originated from AFL-theory.

We remark that other approaches in studying language-theoretic operations may be interpreted in the general (universally algebraic) framework sketched above. E.g. in [7] the power or operators relevant in subregular language families is considered, and in [20] full semi-AFL's provided with several additional operations are investigated.

This paper is organized as follows. The present introduction is followed by four other sections and an appendix. In Section 2 we recall some terminology and basic results concerning monoids generated by closure operators. In Section 3 we determine the structure of monoids generated by closure operstors defining prequasoids (i.e. nontrivial families closed under finite substitution and intersection with regular sets) and pseudoids (i.e. families including all singleton languages and closed under homomorphism and intersection with regular sets). Monoids generated by closure operators defining several full AFL-structures (as considered in [2]) are investigated in Section 4. Section 5 contains some concluding remarks. The appendix is devoted to a (rather long) proof of a lemma in Section 3.

This report has to be considered as a continuation of [2]. Therefore we assume the reader to be familiar with the main definitions and notations of

language-theoretic operators in [2] and with the main results in [2] (In particular Lemma 2.2 and Theorems 3.4, 4.2 and 4.3). With respect to monoids and (pre-)closure operators this paper is (almost) self-contained. Actually we only need elementary notions which can also be found in the introductory chapters of [6, 8, 11, 19, 23].

2. Monoids Generated By Closure Operators

We first recall some elementary terminology and facts from the theory of (partially ordered) monoids. Then we consider (pre-)closure operators and the partially ordered monoids that they generate.

Let U be a set and let $\cdot$ be a binary operation on U. The operation is associative [idempotent, commutative] if for all $a,b,c \in U$ we have $a(bc) = (ab)c$ [$a^2 = a$, $ab = ba$ respectively]. As usual we omit all occurrences of $\cdot$ and we simply write e.g. ab and a^2 instead of $a \cdot b$ and $a \cdot a$ repectively.

A monoid $(U,\cdot,e)$ - or U when $\cdot$ and e are understood - is a set U provided with a binary associative operation, containing a (unique) unit e, i.e. an element e satisfying $ea = ae = a$ for all a in U. An idempotent i in a monoid U is an element satisfying $i^2 = i$, whereas a zero z is an element with the property that $az = za = z$ for all a in U. Thus a unit and a zero are a particular kind of idempotents. A monoid may have several idempotents but has just one unit and at most one zero.

A monoid $(U,\cdot,e)$ is called abelian or a commutative monoid if the operation $\cdot$ is commutative. A (join- or upper-) semilattice (with unit) is an abelian monoid in which each element is idempotent. In a semilattice $(U,\vee,e)$ the monoid operation $\vee$ induces a partial order $\leq$ in U defined by $a \leq b$ iff $a \vee b = b$ for all a,b in U. With respect to this partial order $a \vee b$ is the least upper bound of a and b.

As a special case of a partially ordered monoid [11, 6, 28] we introduce the following concept.

Definition 2.1. A bounded partially ordered monoid or bpo-monoid is an algebra $(U,\cdot,e,z,\leq)$ satisfying

(1) $(U,\cdot,e,z)$ is a monoid with unit e and zero z.

(2) $(U,\leq)$ is a partially ordered set.

(3) $a \leq b$ implies $ac \leq bc$ and $ca \leq cb$, for all $a,b,c \in U$.

(4) $e \leq a \leq z$, for all $a \in U$. □

In partially ordered monoids the order relation is usually defined dually (viz. by means of $z \leq a$ for all a). For our purposes the present order is however more natural. The adjective "bounded" refers to the additional requirement that $e \leq a$ for all a [28].

In the following lemma we mention a few elementary properties of bpo-monoids.

Lemma 2.2. Let $(U,\cdot,e,z,\leq)$ be a bpo-monoid.

(1) $c \leq bc$ and $c \leq cb$ for all $b,c \in U$.

(2) If $c^2 \leq c$ for all c in U, then $(U,\cdot,e)$ is a semilattice with $a \leq b$ iff $ab = b$, for all $a,b \in U$, i.e. the partial order induced by the semilattice $(U,\cdot,e)$ coincides with the original partial order in $(U,\cdot,e,z,\leq)$.

Proof. (1) By 2.1(4) we have $e \leq b$. So 2.1(3) implies $c = ec \leq bc$ and similarly $c \leq cb$.

(2) From (1) we obtain for $b = c$ that $c \leq c^2$. Thus $c^2 \leq c$ implies that each element is idempotent.

If $a \leq b$, then by (1) and 2.1(3), $b \leq ab \leq b^2 = b$. Conversely, if $ab = b$, then by (1), $a \leq ab = b$. Hence $ab = b$ iff $a \leq b$ for all $a,b \in U$. So since $a \leq ba$ we have $aba = ba$. With $b = b^2$ this yields $abba = ba$, or $ab \leq ba$.

Symmetrically we obtain $ba \leq ab$. Hence $\cdot$ is commutative and $(U,\cdot,e)$ is a semilattice. □

A finite monoid U can be completely represented by its Cayley table or multiplication table consisting of all possible products ab for $a,b \in U$. The structure of a finite partially ordered monoid (and in particular of a finite bpo-monoid) is fully given by the pair (C,D), where C is the Cayley table of the monoid and D is the diagram representing the partial order. It follows from Lemma 2.2(2) that for a bpo-monoid in which each element is idempotent either C or D alone fully describes the structure of the bpo-monoid.

We now turn to (pre-)closure operators. Let T be a partially ordered set. A mapping $\Gamma : T \to T$ satisfying (1) $a \leq \Gamma(A)$, and (2) $a \leq b$ implies $\Gamma(a) \leq \Gamma(b)$, for all $a,b \in T$, is called a pre-closure operator on T. A pre-closure operator Γ on T is a closure operator on T if (3) $\Gamma\Gamma(a) = \Gamma(a)$ for each a in T. An operator on T satisfying (1), (2) or (3) is called extensive, monotonic or idempotent respectively.

Since the composition of two pre-closure operators yields again a pre-closure operator and as this composition is also associative, each set of pre-closure operators on T generates a monoid of pre-closure operators with the identity operator $I(a) = a$, for each a in T, as unit. A similar statement does not hold in general for closure operators: the composition of two closure operators is of course still a pre-closure operator, but in general it fails to be a closure operator. It is straightforward to show that each element in a monoid, generated by a set of closure operators, is indeed a closure operator if this monoid is commutative. Moreover each idempotent in a monoid generated by pre-closure operators is obviously a closure operator.

The partial order in T induces a partial order in each monoid of pre-closure operators on T, viz. by definition $\Gamma_1 \leq \Gamma_2$ if $\Gamma_1(a) \leq \Gamma_2(a)$ for all a in T [25]. So for each pre-closure operator Γ on T we have $I \leq \Gamma$ because

$I(a) = a \leq \Gamma(a)$ (cf. 2.1(4)).

Let A_0 be a set and let $P(A_0)$ denote the power set of A_0. Obviously, $P(A_0)$ is partially ordered by set-theoretical inclusion. A nonempty family $\{B_i \mid i \in I, B_i \subseteq A_0\}$ in $P(A_0)$ is called directed if for each i and j in I, there exists a k in I such that $B_i \cup B_j \subseteq B_k$. Let (A_0,Ω_0) be an algebra where A_0 is the carrier set and Ω_0 the set of (finitary) operations. For each subset B of A_0, let $\Gamma_0(B)$ be the smallest subalgebra of A_0 including B. It is well known that Γ_0 is a closure operator on $P(A_0)$, and even a so-called algebraic closure operator: a closure operator Γ_0 on $P(A_0)$ is algebraic if $\Gamma_0(\bigcup_{i\in I} B_i) = \bigcup_{i\in I} \Gamma_0(B_i)$ for each directed family $\{B_i \mid i \in I\}$ of subsets of A_0 [6, 22, 25].

Let $\{\Omega_1,\ldots,\Omega_n\}$ be a finite collection of subsets of Ω_0 such that $\bigcup_{i=1}^{n}\Omega_i = \Omega_0$ and let Γ_i be the algebraic closure operator corresponding to Ω_i $(0 \leq i \leq n)$.

Consider the monoid $Mn\{\Gamma_0,\Gamma_1,\ldots,\Gamma_n\}$ generated by $\{\Gamma_0,\Gamma_1,\ldots,\Gamma_n\}$. If G, G_1 and G_2 are in $Mn\{\Gamma_0,\Gamma_1,\ldots,\Gamma_n\}$ then it easily follows that $G_1 \leq G_2$ implies $G_1G \leq G_2G$ and $GG_1 \leq GG_2$. Since also $G\Gamma_0 = \Gamma_0G = \Gamma_0$ and $G \leq \Gamma_0$ for each G in $Mn\{\Gamma_0,\Gamma_1,\ldots,\Gamma_n\}$, the monoid $Mn\{\Gamma_0,\Gamma_1,\ldots,\Gamma_n\}$ is a bpo-monoid with unit I and zero Γ_0. As $\Gamma_0,\Gamma_1,\ldots,\Gamma_n$ are algebraic closure operators it can be shown that $\Gamma_0(A) = (\Gamma_1\ldots\Gamma_n)^*(A) = \bigcup_{i=0}^{\infty}(\Gamma_1\ldots\Gamma_n)^i(A)$ [25]. Moreover, we have that $A = \Gamma_0(A)$ iff $A = \Gamma_1\ldots\Gamma_n(A)$ [25]. In both these statements we may also take any other permutation of $\Gamma_1,\ldots,\Gamma_n$.

Since $(\Gamma_1\ldots\Gamma_n)^i \leq (\Gamma_1\ldots\Gamma_n)^{i+1}$ by Lemma 2.2(1), it is also straightforward to show that $Mn\{\Gamma_0,\ldots,\Gamma_n\}$ is finite iff there exists an $i \geq 0$ such that $\Gamma_0(A) = (\Gamma_1\ldots\Gamma_n)^i(A)$ for each $A \subseteq A_0$. Note that the expression $\Gamma_0 = (\Gamma_1\ldots\Gamma_n)^i$ is an example of what is called a defining relation in semigroup theory [8, 19, 23]. A defining relation of the form $\Gamma_0 = (\Gamma_1\ldots\Gamma_n)^i$ implies that $Mn\{\Gamma_0,\Gamma_1,\ldots,\Gamma_n\}$ is also generated by $\{\Gamma_1,\ldots,\Gamma_n\}$, i.e. it

equals $Mn\{\Gamma_1,\ldots,\Gamma_n\}$.

Finally, we remark that several results have been established on monoids generated by idempotents [18, 21]. They are however not applicable to the cases in which we are interested.

3. Prequasoids and Pseudoids

In this section we describe the structure of bpo-monoids generated by closure operators related to prequasoids and to pseudoids. For the relevance of these structures in language theory we refer to [1, 2, 5] and [4, 5] respectively, and the references mentioned there.

Let ALL be the family of all languages over finite subsets of the arbitrary but fixed countable set of symbols Σ_ω. Remember that a σ-simple family [2] is a family including SYMBOL = $\{\{\alpha\} \mid \alpha \in \Sigma_\omega\}$ and closed under isomorphism (i.e. renaming of symbols or, equivalently, one-to-one SYMBOL-substitution). By ALL [σ-SIMPLE] we denote the class of all [σ-simple] language families.

Obviously, ALL equals P(ALL) and it is easy to show that operators like Θ, Φ, Δ, Π, $\hat{\mathcal{P}}$, $\hat{\mathfrak{F}}$, $\hat{\mathcal{H}}$, etc. are algebraic closure operators on ALL (cf. [3]). In order to avoid trivialities we restrict however the domain of the operators to σ-SIMPLE. Clearly, σ-SIMPLE is not a power set although it possesses the following properties: (i) it is closed under (arbitrary) union and (arbitrary) intersection, (ii) each family in σ-SIMPLE is the (arbitrary) union of finitely generated σ-simple families. It is well-known [6, 22, 3] that without loss of generality we may replace $P(A_0)$ by any class satisfying (i) and (ii), in the last part of Section 2 (Even weaker properties than (i) and (ii) suffice, as abstracted in the notion of algebraic or compactly generated lattice; [6, 22, 3]). It is now straightforward to show that also operators like M, S, F, etc. are algebraic closure operators (on σ-SIMPLE).

	I	Φ	Δ	ΦΔ	ΔΦ	ΔΦΔ	Π
I	I	Φ	Δ	ΦΔ	ΔΦ	ΔΦΔ	Π
Φ	Φ	Φ	ΦΔ	ΦΔ	Π	Π	Π
Δ	Δ	ΔΦ	Δ	ΔΦΔ	ΔΦ	ΔΦΔ	Π
ΦΔ	ΦΔ	Π	ΦΔ	Π	Π	Π	Π
ΔΦ	ΔΦ	ΔΦ	ΔΦΔ	ΔΦΔ	Π	Π	Π
ΔΦΔ	ΔΦΔ	Π	ΔΦΔ	Π	Π	Π	Π
Π	Π	Π	Π	Π	Π	Π	Π

Table I.

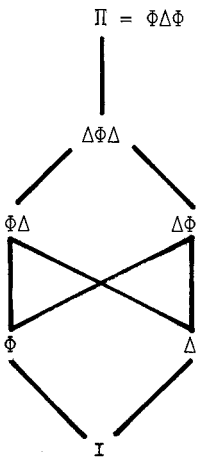


Diagram 1.

For pre-closure operators Γ_1 and Γ_2 we write as usual $\Gamma_1 < \Gamma_2$ if $\Gamma_1 \leq \Gamma_2$ and $\Gamma_1 \neq \Gamma_2$ (i.e. there exists a σ-simple family K such that $\Gamma_1(K) \subset \Gamma_2(K)$). The operators Γ_1 and Γ_2 are called incomparable if neither $\Gamma_1 \leq \Gamma_2$ nor $\Gamma_2 \leq \Gamma_1$.

The first monoid that we consider is the one generated by Φ and Δ. These operators together define the concept of prequasoid (Let $\Omega_1 = \{f \mid f$ is a finite substitution$\}$, $\Omega_2 = \{ \cap R \mid R$ is regular$\}$, and $\Omega_0 = \Omega_1 \cup \Omega_2$. Then $\Gamma_1 = \Phi$, $\Gamma_2 = \Delta$, $\Gamma_0 = \Pi = \Phi\Delta\Phi$; cf. [2] Lemma 2.2). As a main tool in determining the structure of $Mn\{\Phi,\Delta\}$ we need the following lemma.

Lemma 3.1. $\Delta\Phi\Delta < \Pi$. □

The proof of this result is rather long and therefore postponed to the Appendix.

Theorem 3.2. $Mn\{\Phi,\Delta\}$ is a bpo-monoid of 7 elements with unit I and zero Π. Moreover, (i) I, Π, Δ and Φ are the idempotents, (ii) the multiplication is given in Table I, (iii) the partial order is given in Diagram 1.

Proof. First, (i) is straightforward, whereas the Cayley table can be easily computed using the defining relation $\Pi = \Phi\Delta\Phi$ ([2] Lemma 2.2). So it remains

	I	Θ	Δ	ΔΘ	Ψ
I	I	Θ	Δ	ΔΘ	Ψ
Θ	Θ	Θ	Ψ	Ψ	Ψ
Δ	Δ	ΔΘ	Δ	ΔΘ	Ψ
ΔΘ	ΔΘ	ΔΘ	Ψ	Ψ	Ψ
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ

Table II.

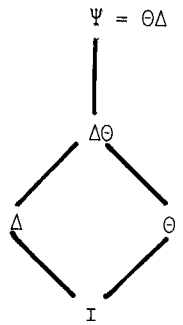


Diagram 2.

to prove (iii). Inspection of this diagram yields that it suffices to prove (apart from Lemma 3.1) that (1) $\Delta\Phi$ and $\Phi\Delta$ are incomparable, and (2) Δ and Φ are incomparable.

Ad(1): Suppose $\Delta\Phi \leq \Phi\Delta$ [$\Phi\Delta \leq \Delta\Phi$], then $\Pi = \Phi\Delta\Phi \leq \Phi\Phi\Delta = \Phi\Delta \leq \Delta\Phi\Delta$

[$\Pi = \Phi\Delta\Phi \leq \Delta\Phi\Phi = \Delta\Phi \leq \Delta\Phi\Delta$], contradicting Lemma 3.1.

Ad(2): Suppose $\Phi \leq \Delta$ [$\Delta \leq \Phi$], then $\Pi = \Phi\Delta\Phi \leq \Delta \leq \Delta\Phi\Delta$

[$\Pi = \Phi\Delta\Phi \leq \Phi \leq \Delta\Phi\Delta$], contradicting Lemma 3.1. □

The corresponding result for pseudoids reads as follows. $\Psi(K)$ denotes the smallest pseudoid including K.

Theorem 3.3. $Mn\{\Theta,\Delta\}$ is a bpo-monoid containing 5 elements with unit I and zero Ψ. Furthermore, (i) I, Ψ, Δ and Θ are the idempotents, (ii) the Cayley table is given in Table II, (iii) the partial order is given in Diagram 2.

Proof. First we show that $\Psi = \Theta\Delta$ for σ-simple families (This fact was mentioned in [4] without proof). Obviously $K \subseteq \Theta\Delta(K) \subseteq \Psi(K)$ for each σ-simple family K, and $\Theta\Delta(K)$ is closed under homomorphism (i.e. $\Theta\Theta\Delta(K) \subseteq \Theta\Delta(K)$ since Θ is idempotent). So it remains to show that $\Theta\Delta(K)$ is closed under Δ. Let $L \in \Delta\Theta\Delta(K)$, i.e. there exist $L_1 \subseteq \Sigma_1^*$ in K, regular languages $R_1 \subseteq \Sigma_1^*$ and $R_2 \subseteq \Sigma_2^*$, and a homomorphism $h : \Sigma_1^* \to \Sigma_2^*$ such that $L = R_2 \cap h(R_1 \cap L_1)$. Then $L = h(h^{-1}(R_2) \cap R_1 \cap L_1)$. Since $h^{-1}(R_2) \cap R_1$ is regular, we obtain $L \in \Theta\Delta(K)$.

Using this defining relation it is straightforward to compute the Cayley table. With respect to the partial order we have $I \leq \Delta \leq \Delta\Theta \leq \Psi$, and similarly $I \leq \Theta \leq \Delta\Theta \leq \Psi$. So in establishing the correctness of Diagram 2, it suffices to show (1) $\Psi \neq \Delta\Theta$, and (2) Δ and Θ are incomparable.

Ad(1): Suppose $\Psi = \Delta\Theta$, then $\Theta\Delta = \Delta\Theta$ and hence $\Theta\Delta\Phi = \Delta\Theta\Phi$. Since $\Theta\Phi = \Phi$ and $\Theta\Delta\Phi = \Pi$ ([2] Lemma 2.2), we obtain $\Pi = \Delta\Phi$, contradicting 3.2.

Ad(2): Suppose $\Theta \leq \Delta$ [$\Delta \leq \Theta$], then $\Psi = \Theta\Delta \leq \Delta$ [$\Psi = \Theta\Delta \leq \Theta$] and hence $\Psi = \Delta\Theta$ [$\Psi = \Delta\Theta$]. But this contradicts (1). □

Note that $Mn\{\Theta,\Delta\}$ is not a semilattice (cf. Lemma 2.2(2)), since $\Delta\Theta\Delta\Theta = \Psi$, i.e. $\Delta\Theta$ is not idempotent.

Finally, we consider the monoid generated by Θ,Φ and Δ.

Theorem 3.4. $Mn\{\Theta,\Phi,\Delta\}$ is a bpo-monoid containing 10 elements with unit I and zero Π, and (i) the idempotents are I,Π,Ψ,Δ,Θ and Φ, (ii) the partial order is given in Diagram 3.

Proof. Apart from obvious inequalities such as $\Psi \leq \Phi\Delta$, $\Theta \leq \Phi$, $\Delta\Theta \leq \Delta\Phi$, etc. we have to show:

(1) Ψ and Φ are incomparable.

Suppose $\Psi \leq \Phi$, then $\Pi = \Psi\Phi \leq \Phi$ ([2] Lemma 2.2) which contradicts Theorem 3.2. On the other hand the supposition $\Phi \leq \Psi$ yields $\Pi = \Psi\Phi \leq \Psi$ and hence $\Pi = \Phi\Pi \leq \Phi\Psi = \Phi\Delta$ contradicting Theorem 3.2.

(2) Ψ and $\Delta\Phi$ are incomparable.

Suppose $\Delta\Phi \leq \Psi$ [$\Psi \leq \Delta\Phi$], then $\Pi = \Psi\Delta\Phi \leq \Psi$ [$\Pi = \Psi\Phi \leq \Delta\Phi\Phi = \Delta\Phi$] which contradicts $\Psi < \Pi$ [Theorem 3.2.] □

In Theorem 3.4 we did not give the Cayley table because its major part was already mentioned in Theorems 3.2 and 3.3, whereas the remaining 30 entries in this Cayley table can directly be determined with the help of $\Psi = \Theta\Delta$, $\Pi = \Theta\Delta\Phi = \Phi\Delta\Phi$ and $\Phi\Theta = \Theta\Phi = \Phi$. Note that $\Delta\Theta \leq \Theta\Delta$ implies $\Phi\Delta\Theta \leq \Phi\Delta$.

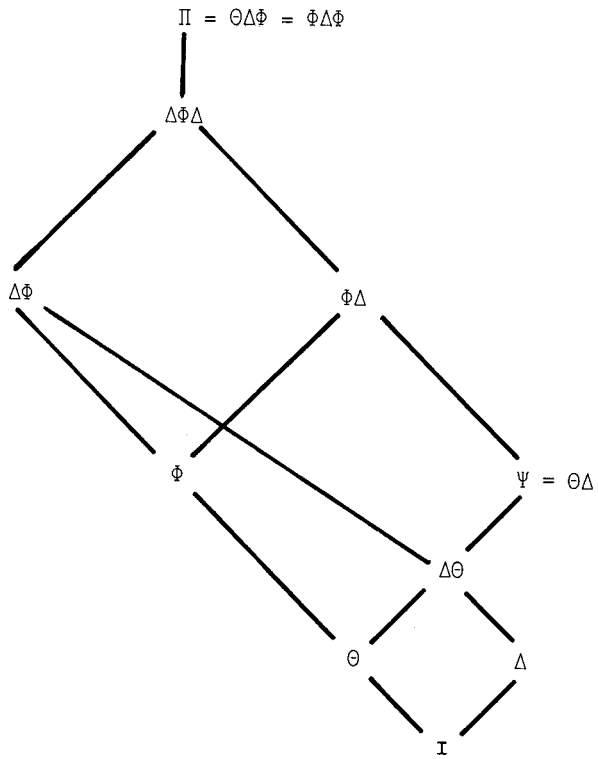


Diagram 3.

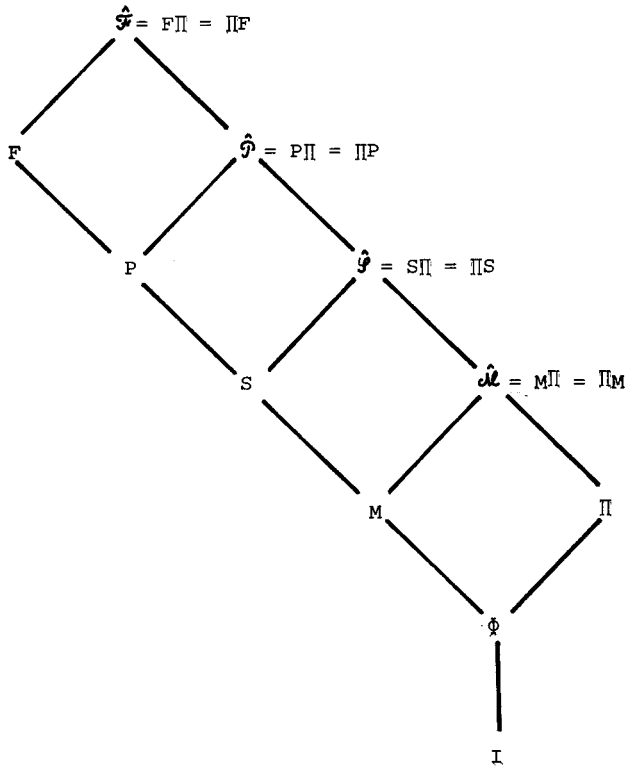


Diagram 4.

4. Full AFL's and Related Structures

In this section we consider monoids generated by operators related to full AFL-structures as surveyed in [2]. The domain of each operator is as in the previous section restricted to σ-SIMPLE.

First we determine the structure of the monoid generated by Φ,Π,M,S,P and F. Recall that $\hat{\mathfrak{X}} = X\Pi = \Pi X$ for $X \in \{M,S,P,F\}$ by [2] Theorems 4.2 and 4.3.

Theorem 4.1. $Mn\{\Phi,\Pi,M,S,P,F\}$ is a bpo-monoid containing 11 elements with unit I and zero $\hat{\mathfrak{F}}$ in which each element is idempotent. Moreover, (i) the partial order is given by Diagram 4, (ii) the multiplication is given by the fact that $\Gamma_1\Gamma_2$ is the least upper bound of Γ_1 and Γ_2 with respect to the partial order $\leq$.

Proof. Using [2] Lemma 2.2 and Theorems 4.2, 4.3 and 4.6 it is straightforward to show that $Mn\{\Phi,\Pi,M,S,P,F\}$ contains no other elements than those in Diagram 4. Hence, by [2] Theorem 4.2, each of its elements is idempotent and so Lemma 2.2(2) is applicable, which proves (ii). It remains to show (i).

The family of metalinear languages [linear context-free languages] is a full P-AFL [full S-AFL] but not a full F-AFL [full P-AFL][27, 12]. The set-theoretical union of two incomparable full F-AFL's is a full M-AFL but not a full S-AFL [15, 2]. The family of finite languages is a pre-quasoid but not a full M-AFL.

It now suffices to disprove $\Pi \leq F$.

Let K_0 be the family of all languages which are either finite or include an infinite regular subset. Define a subset K of K_0 by $K = \{L \mid r(L) \in K_0 \text{ for all regular substitutions } r\}$. Clearly K is σ-simple. Let L_1 and L_2 be in K. Since $r(L_1 \cup L_2) = r(L_1) \cup r(L_2)$ for each (regular) substitution r and K_0 is closed under union, K is closed under union. Similarly, as K_0 is closed under concatenation [Kleene *] and $r(L_1L_2) = r(L_1)r(L_2)$ $[r(L_1^*) = r(L_1)^*]$, K is also closed under concatenation [Kleene *]. Moreover, since the composition of two regular substitutions yields again a regular substitution, K is also closed under regular substitution. Hence $F(K) = K$.

We now show that $K \subset \Pi(K)$. Let $L = \{a^nb^n \mid n \geq 1\} \cup a^*$. It is straightforward to prove that L is in K. Then $L \cap a^+b^+ = \{a^nb^n \mid n \geq 1\}$ is in $\Pi(K) - K$, because $L \cap a^+b^+$ does not include an infinite regular subset.

This proves (i) and the theorem. □

We now consider the monoid generated by Π,R,A and H. In order to avoid trivialities we restrict the domain of each operator to be considered to <u>α-SIMPLE</u>, i.e. the class of all α-simple families. Recall that a family is α-simple [2] if it includes SYMBOL and it is closed under isomorphism and under union with SYMBOL-languages. <u>α-SIMPLE</u> also fails to be a power set but

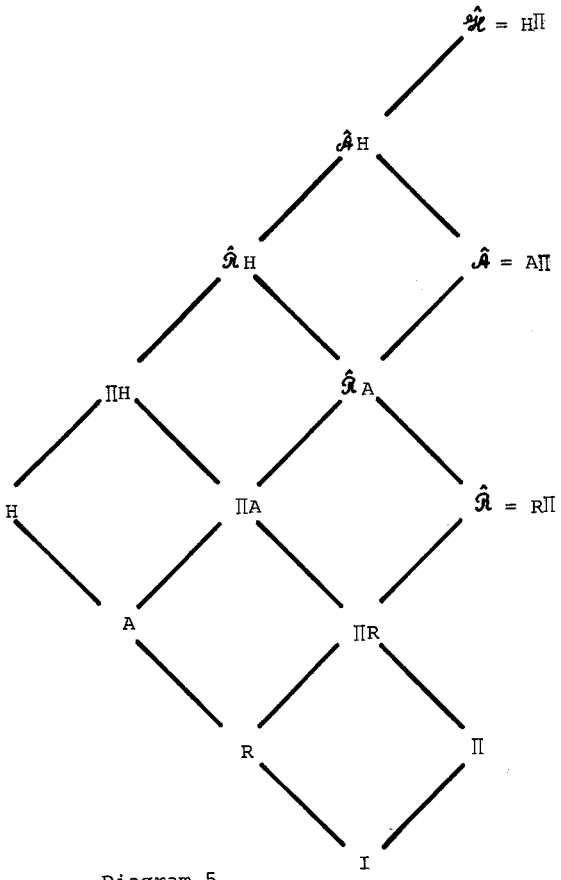


Diagram 5.

it meets the conditions (i) and (ii) mentioned in the beginning of Section 3, and therefore the last part of Section 2 also remains valid in this case. Moreover, the operators R, A and H turn out to be algebraic closure operators on α-SIMPLE (cf. proofs of [1] Theorem 4.4 and [2] Theorem 3.4). Clearly, ALPHA = $\{\Sigma \mid \Sigma \subseteq \Sigma_\omega, \Sigma \text{ is finite}\}$ is the smallest α-simple family. Note that ALPHA is hyper-algebraically closed, i.e. H(ALPHA) = ALPHA, and hence ALPHA is also a fixed point of A and R.

Theorem 4.2. Mn{Π,R,A,H} is a bpo-monoid consisting of 14 elements with unit I and zero $\hat{\mathcal{K}}$, and (i) the idempotents are I, $\hat{\mathcal{K}}$, $\hat{\mathcal{A}}$, $\hat{\mathfrak{R}}$,H,A,R and Π, (ii) the partial order is given in Diagram 5.

Proof. We already remarked that R,A and H are idempotent on α-SIMPLE. With respect to the partial order, inspection of Diagram 5 yields that it suffices to disprove the following (in)equalities: (1) $\hat{\mathcal{H}} = \hat{\mathcal{A}}H$; (2) $\hat{\mathcal{A}} \leq \hat{\mathcal{R}}H$; (3) $\hat{\mathcal{R}} \leq \Pi H$; (4) $\Pi \leq H$; (5) $H \leq \hat{\mathcal{A}}$; (6) $A \leq \hat{\mathcal{R}}$; and (7) $R \leq \Pi$.

Ad(1): $\hat{\mathcal{H}}$(ALPHA) = ETOL ⊃ CF = $\hat{\mathcal{A}}H$(ALPHA)

Ad(2): $\hat{\mathcal{A}}$(ALPHA) = CF ⊃ REG = $\hat{\mathcal{R}}H$(ALPHA)

Ad(3): $\hat{\mathcal{R}}$(ALPHA) = REG ⊃ FIN = ΠH(ALPHA)

Ad(4): Π(ALPHA) = FIN ⊃ ALPHA = H(ALPHA).

For the remaining cases we note that CF [REG,FIN] is a full A-AFL [full R-AFL, prequasoid] but not a full H-AFL [full A-AFL, full R-AFL]. □

The Cayley table of Mn{Π,R,A,H} can be computed in a straightforward way using Theorem 4.2(i) and [2] Theorem 3.4.

In connection with inequality (4) in the proof we remark that there exists a less trivial fixed point of H which could also serve as a counter-example. Viz. in [29] it was shown that the family of languages generated by "λ-free unconditional transfer context-free programmed grammars under rightmost interpretation" is a fixed point of H but not of Π.

5. Concluding Remarks

In this paper we investigated a few monoids generated by algebraic closure operators originating from language theory in order to study (i) (in)dependency of sets consisting of language-theoretic operations, and (ii) the relative power of successively applying these closure operators.

Consider as an example Ω_1 and Ω_2 as defined in Section 3. From Theorem 3.2 it follows that Ω_1 and Ω_2 are independent, and that e.g. the processes of closure under Ω_1 and closure under Ω_2 are not interchangeable, i.e. $\Phi\Delta$ and $\Delta\Phi$ are incomparable. We leave all other conclusions of this kind (from Theorems 3.2 - 3.4, 4.1 and 4.2) to the reader.

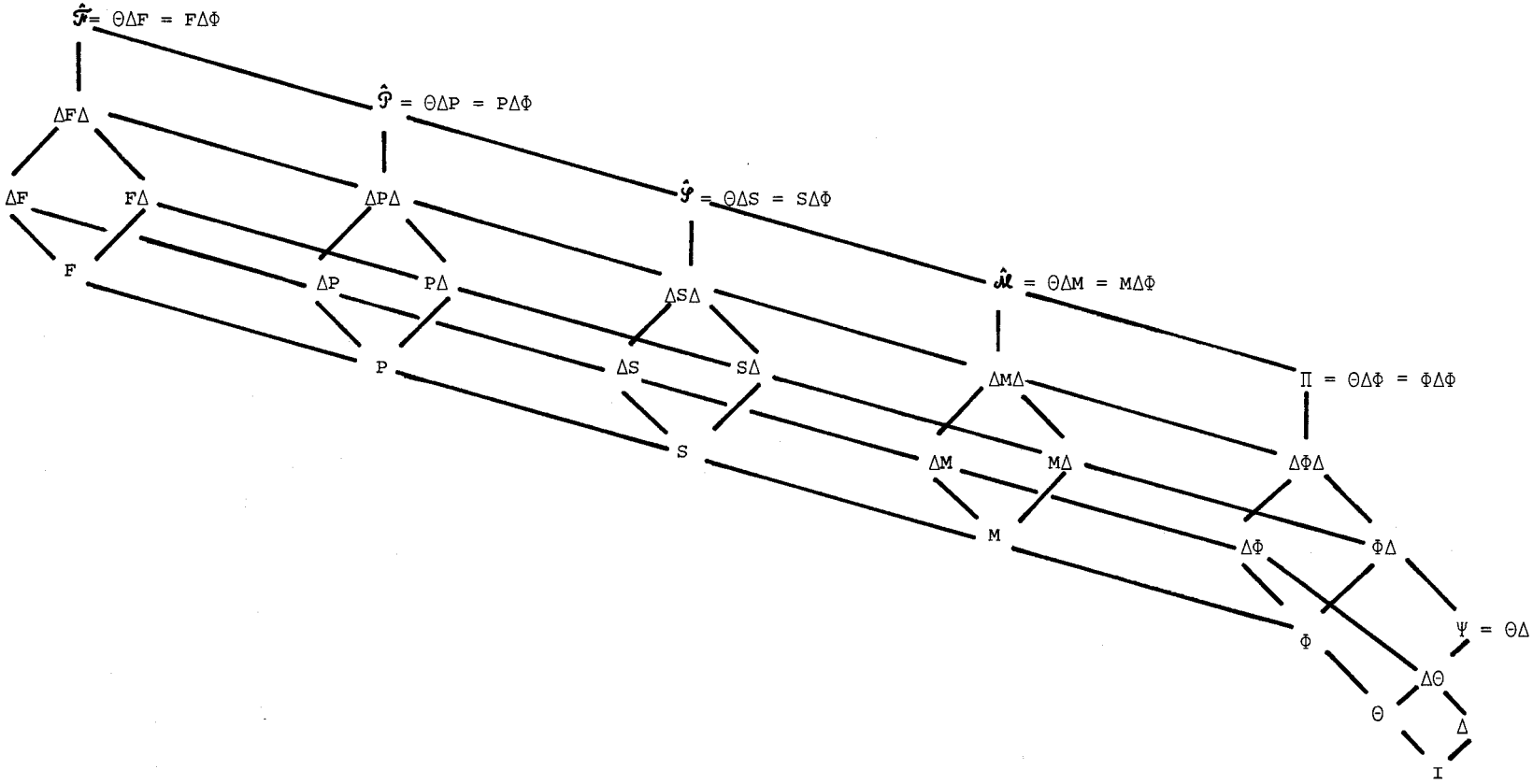

$\hat{\mathfrak{F}} = \Theta\Delta F = F\Delta\Phi$
$\hat{\mathfrak{P}} = \Theta\Delta P = P\Delta\Phi$
$\hat{\mathfrak{S}} = \Theta\Delta S = S\Delta\Phi$
$\hat{\mathfrak{M}} = \Theta\Delta M = M\Delta\Phi$
$\Pi = \Theta\Delta\Phi = \Phi\Delta\Phi$
$\Psi = \Theta\Delta$
$\Delta F\Delta$
ΔF
$F\Delta$
$\Delta P\Delta$
F
ΔP
$P\Delta$
$\Delta S\Delta$
P
ΔS
$S\Delta$
$\Delta M\Delta$
S
ΔM
$M\Delta$
$\Delta\Phi\Delta$
M
$\Delta\Phi$
$\Phi\Delta$
Φ
$\Delta\Theta$
Θ
Δ
I

Diagram 6.

We emphasize that we only considered a few specific collections $\{\Omega_1,\ldots,\Omega_n\}$ (or $\{\Gamma_1,\ldots,\Gamma_n\}$) and that many other combinations are possible. Take for instance the monoid generated by Θ,Φ,Δ,M,S,P and F. If there exists a σ-simple language family K such that $\Pi(K) - \Delta F\Delta(K) \neq \emptyset$ (as we conjecture but are not able to prove), then this bpo-monoid contains 30 elements, whereas its partial order is as given in Diagram 6 (The 900 entries in the Cayley table are easy to compute using [2] Lemma 2.2 and Theorems 4.2 - 4.3).

Or to consider another combination, it can also be shown that $Mn\{\Psi,\eta\}$, where η is the dhyper-algebraic extension as studied in [4], contains 5 elements and that this bpo-monoid is isomorphic with $Mn\{\Delta,\Theta\}$ (Ψ [η] corresponds to Δ [Θ]; the domain of Ψ and η is α-SIMPLE).

Finally, one could also study monoids generated by language-theoretic pre-closure operators (For pre-closure operators on language families we refer to [30, 31, 32]). Consider e.g. $Mn\{\Pi,H_1\}$where H_1 is the 1-restricted hyper-algebraic extension [2, 30] (The domain of the operators is α-SIMPLE). In order to obtain a bpo-monoid we have to add H_1^* to the generating set; then it can be shown that $Mn\{\Pi,H_1,H_1^*\}$ is an infinite bpo-monoid with unit I and zero $\mathcal{Z}_1 = H_1^*\Pi$, of which the partial order is as in Diagram 7. (The hardest part of the argument - viz. the infiniteness of $Mn\{\Pi,H_1,H_1^*\}$ - follows from results in [30]).

Appendix

In this appendix we present a proof of Lemma 3.1, i.e. we show that there exists a σ-simple family K and a language M such that $M \in \Phi\Delta\Phi(K)$ and $M \notin \Delta\Phi\Delta(K)$.

Let K consist of all SYMBOL-languages and all isomorphic copies of $L = \{a^n b^{n+1} \mid n \geq 2\}$. Then K is σ-simple. Define M by $M = g(f(L) \cap R)$ with $f(a) = \{a\}$, $f(b) = \{b,c,d\}$, $R = \{a^{2m}b^nc \mid m,n \geq 1\} \cup \{a^{2m+1}b^nd \mid m,n \geq 1\}$, and $g(\alpha) = \{\alpha,\alpha\alpha\}$ for α in $\{a,b\}$, $g(\alpha) = \{\alpha\}$ for α in $\{c,d\}$. Clearly, the

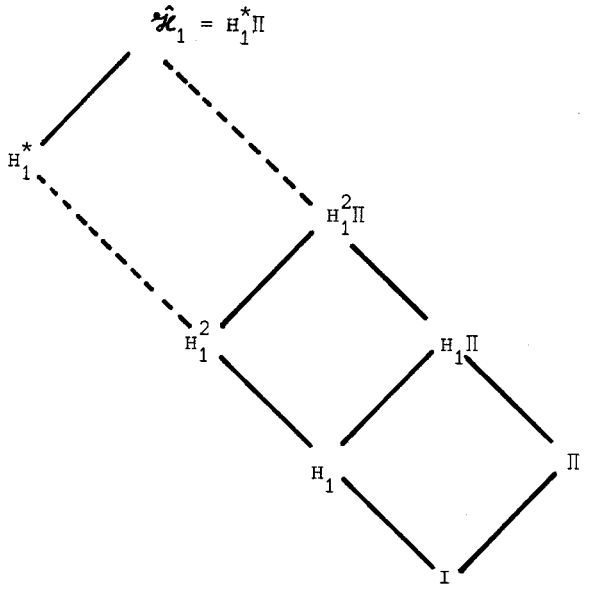


Diagram 7.

following holds: $M = \{a^k b^\ell c \mid \exists n\colon n \le k,\ell \le 2n$ and n is even$\} \cup \{a^k b^\ell d \mid \exists n\colon n \le k,\ell \le 2n$ and n is odd$\} \in \Phi\Delta\Phi(K)$.

So we have to prove that $M \notin \Delta\Phi\Delta(K)$. Suppose to the contrary that $M \in \Delta\Phi\Delta(K)$, then we may assume that $M = \phi(L \cap S) \cap T$ where S and T are regular sets and ϕ is a finite substitution. We will derive a contradiction.

Let $\Sigma = \{a,b\}$, $V = \Sigma \cup \{c,d\}$, $\phi(a) = A$ and $\phi(b) = B$. It may obviously be assumed that $A,B \subseteq a^*b^*\{\lambda,c,d\}$.

Observation 1: B contains words $b^t c$ and $b^s d$ $(t,s \ge 0)$.

Suppose to the contrary that for instance B does not contain $b^t c$ (the argument for $b^s d$ is analogous). Then obviously A contains a word $b^t c$, B contains λ, and there exists $p,q \ge 1$ such that a^p and b^q are in A. Let k be an even number such that $kp \ge r_o$ where r_o equals the number of states of the deterministic finite state acceptor which recognizes the regular set T. From $a^n b^{n+1}$ in $L \cap S$ we can then obtain by ϕ a word $w = a^{kp} b^{(n-1-k)q} b^t c$ using that

a^p, b^q, $b^t c \in A$ and $\lambda \in B$. Choose n sufficiently large such that $(n-1-k)q + t > 2kp$ (Note that $L \cap S$ has to be infinite). Then w is in $\phi(L \cap S) - M$ and consequently $w \notin T$. Moreover the language $V^* - T$ can be recognized by a finite automaton A with r_o states (cf. [10]). During the processing of the b's in the word w, A must have passed through the same state several times (cf. [10]). Since $r_o \leq kp$, this means that there exists a word $w' = a^{kp} b^v c$ in $V^* - T$ with $kp \leq v \leq 2kp$ (We will use this proof technique again in the sequel and we will simply refer to it as "shortening" instead of repeating the complete argument). But w' is in M, contradicting the fact that $w' \notin T$.

Observation 2: There exist numbers p and q ($p,q \geq 1$) with $a^p \in A$ and $b^q \in B$.

Suppose not. Then we consider the following cases

(i) Let for all $p,q \geq 1$, $a^p \notin A$ and $b^q \notin B$. But then we only obtain by ϕ from L words like $a^k b^\ell \mu$ (μ equals c or d), where k and ℓ are bounded by two times the maximum over i and j such that $a^i b^j \{\lambda, c, d\}$ is in $A \cup B$.
This contradicts the fact that M is infinite.

(ii) There exists a $p \geq 1$ with $a^p \in A$, and for all $q \geq 1$: $b^q \notin B$. Then obviously there exists a $q \geq 1$ such that $b^q \in A$. By an analogous argument as for Observation 1 we are then able to derive a contradiction.

(iii) There exists a $q \geq 1$ with $b^q \in B$, and for all $p \geq 1$: $a^p \in A$. The argument is similar as in the previous case.

Observation 3: $A \cap b^* = \emptyset$ and $B \cap a^* = \emptyset$.

Suppose b^j is in A with $j \geq 0$. Let a^p be in A and b^q in B ($p,q \geq 1$; cf. Observation 2). We take the number k sufficiently large such that $kp \geq r_o$ (cf. Observation 1). From $a^n b^{n+1}$ in $L \cap S$ we can obtain by ϕ a word $w = a^{kp} b^{(n-k)j+nq+v} \mu$ (v equals one of the numbers t or s from Observation 1; $\mu \in \{c, d\}$) using the fact that a^p an b^j are in A. We choose n sufficiently large such that $(n-k)j + nq + v > 2kp$. Then w is in $\phi(L \cap S) - M$ and hence

not in T. By "shortening" there exists however a word $w' = a^{kp}b^{i}\mu$ in $V^* - T$ with $kp \leq i \leq 2kp$. Since w' is in both $V^* - T$ and M we obtain a contradiction.

Observation 4: $a \in A$ and $b,c \in B$.

Since $\lambda \notin A$ and $\lambda \notin B$ (cf. Observation 3) the word a^2b^2c of M can only be obtained by ϕ from the word a^2b^3 of L. This is only possible when a is in A and b, c are in B.

Observation 5: There exists a number $p \geq 2$ such that $a^p \in A$ or there exists a number $q \geq 2$ such that $b^q \in B$.

Suppose the contrary, i.e. let the only words over Σ which are contained in A (B) be a (b) and words like a^ib^j $(i,j \geq 1)$. Then it is easy to see that the absolute difference of the number of a's and the number of b's in $\phi(L) \cap a^*b^*\{\lambda,c,d\}$ is bounded. However in M this difference is unbounded. This contradicts $M \subseteq \phi(L) \cap a^*b^*\{\lambda,c,d\}$.

Now we are able to show that $M = \phi(L \cap S) \cap T$ yields a contradiction. Suppose $M = \phi(L \cap S) \cap T$. Then $b^q \in B$ for some $q \geq 2$ (Observation 5; the case $a^p \in A$ is analogous). Since $L \cap S$ is infinite there exists a number $k \geq r_o + 1$ with $a^kb^{k+1} \in L \cap S$. We distinguish the following two cases:
(i) k is odd. Then there exists a word $w = a^kb^{kq+t}c \in \phi(L \cap S) - M$ (cf. Observation 1) and consequently $w \in V^* - T$. By "shortening" we obtain a word $w' = a^kb^ic \in V^* - T$ with $k - 1 \leq i \leq 2k - 2$. But then $k - 1 \leq k,i \leq 2k - 2$ holds and k -1 is even, which implies that w' is in M. This contradicts $w' \notin T$.
(ii) k is even. Hence there is a word $u = a^kb^{kq+s}d \in \phi(L \cap S) - M$ (cf. Observation 1) and u is in $V^* - T$. By "shortening" we obtain a word $u' = a^kb^jd \in V^* - T$, where $k - 1 \leq j \leq 2k - 2$. Then clearly $k - 1 \leq k,j \leq 2k - 2$ holds and k - 1 is odd, which implies that u' is in M. But this yields a contradiction because $u' \notin T$.

This completes the proof of Lemma 3.1.

References

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

2. P.R.J. Asveld, Extensions of language families and canonical forms for full AFL-structures, TW-memorandum No. 167, Twente University of Technology, Enschede, The Netherlands (1977).

3. P.R.J. Asveld, Incomparable elements in algebraic lattices with an application to AFL-theory, TW-memorandum No. 202, Twente University of Technology, Enschede, The Netherlands (1978).

4. P.R.J. Asveld, J. Engelfriet, Iterated deterministic substitution, Acta Informatica 8 (1977) 285-302.

5. P.R.J. Asveld, J. Engelfriet, Extended linear macro grammars, iteration grammars, and register programs, TW-memorandum No. 209, Twente University of Technology, Enschede, The Netherlands (1978).

6. G. Birkhoff,"Lattice Theory", 3rd ed., Amer. Math. Soc., Providence, R.I. (1967).

7. J.A. Brzozowski, Hierarchies of aperiodic languages, Revue Française Automat. Inform. Rech. Opérat. IT 10 (1976) 33-49.

8. A.H. Clifford, G.B. Preston,"The Algebraic Theory of Semigroups", Amer. Math. Soc., Providence, R.I., Vol. 1 (1961), Vol. 2 (1967).

9. S. Comer, J. Johnson, The standard semigroup of operators of a variety, Algebra Universalis 2 (1972) 77-79.

10. S. Eilenberg,"Automata, Languages and Machines", Academic Press, New York, Vol. A (1974).

11. L. Fuchs,"Partially Ordered Algebraic Systems", Pergamon Press, Oxford (1963).

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

13. S. Ginsburg, S.A. Greibach, Abstract families of languages, pp. 1-32 in [14].

14. S. Ginsburg, S.A. Greibach, J.E. Hopcroft, Studies in abstract families of languages, Mem. Amer. Math. Soc. 87 (1969).

15. S. Ginsburg, E.H. Spanier, On incomparable Abstract Families of Languages (AFL), J. Comp. System Sci. 9 (1974) 88-108.

16. G. Grätzer, H. Lakser, The structure of pseudocomplemented distributive lattices II: congruence extension and amalgamation, Trans. Amer. Math. Soc. 156 (1971) 343-358.

17. S.A. Greibach, J.E. Hopcroft, Independence of AFL operations, pp. 33-40 in [14].

18. H. Höft, A normal form for some semigroups generated by idempotents, Fund. Math. 84 (1974) 75-78.

19. J.M. Howie,"An Introduction to Semigroup Theory", Academic Press, London (1976).

20. S.Y. Itoga, Comparing language operations, Math. Systems Th. 10 (1977) 305-321.

21. A. Iwanik, A remark on a paper of H. Höft, Fund. Math. 84 (1974) 79-80.

22. B. Jónsson,"Topics in Universal Algebra", Lecture Notes in Mathematics 250, Springer, Berlin-Heidelberg-New York.

23. E.S. Ljapin,"Semigroups", Amer. Math. Soc.,Providence, R.I. (1963).

24. E. Nelson, Finiteness of semigroups of operators in universal algebra, Can. J. Math. 19 (1967) 764-768.

25. R.S. Pierce, Closure spaces with an application to ring theory, pp. 565-616 in "Lectures on Rings and Modules", Lecture Notes in Mathematics 246, Springer, Berlin-Heidelberg-New York (1972).

26. D. Pigozzi, On some operations on classes of algebras, Algebra Universalis 2 (1972) 346-353.

27. A. Salomaa,"Formal Languages", Academic Press, New York (1973).

28. Z. Shmuely, On bounded po-semigroups, Proc. Amer. Math. Soc. 58 (1976) 37-43.

29. J. van Leeuwen, A study of complexity in hyper-algebraic families, pp. 323-333 in: A. Lindenmayer, G. Rozenberg (eds.),"Automata, Languages, Development", North-Holland, Amsterdam (1976).

30. J. Engelfriet, Iterating iterated substitution, Theor. Comp. Sci. 5 (1977) 85-100.

31. S.A. Greibach, Chains of full AFL's, Math. Systems Th. 4 (1970) 231-242.

32. S.A. Greibach, Syntactic operators on full semi-AFL's, J. Comp. System Sci. 6 (1972) 30-76.

P A R T I I I

LATTICES OF FIXED POINTS

Chapter 7

Incomparable Elements in Algebraic Lattices with an Application to AFL-theory

Reprinted from *TW-memorandum No. 202 (January 1978)*

Abstract

Two special types of algebraic (or compactly generated) lattices, called GG- and GH-lattices, are introduced. These lattices are uncountable, and each element in these lattices (apart from the zero and unit) has an incomparable element. The main results characterize those elements which have a largest incomparable element. Then two particular kinds of algebras, called GG- and GH-algebra, are defined and it is shown that the lattice of subalgebras of a GG-algebra [GH-algebra] is a GG-lattice [GH-lattice]. Finally, some applications to the theory of Abstract Families of Languages (or AFL) are discussed.

1. Introduction

An important tool for investigating the structure of an algebra is the description of the class of all its subalgebras. A classic result in universal algebra says that the class of subalgebras Sa(A) of a given algebra A constitutes a so-called algebraic or compactly generated lattice (with the set-theoretical inclusion as partial order) in which the finitely generated subalgebras are precisely the compact elements [6, 7, 19, 20, 26] (cf. Theorem 2.8 below. Recall that an element a of a lattice L is [strongly] compact if, whenever $a \leq \bigvee S$ for some nonempty subset S of L, there exists a finite subset F of S [an element b of S] with $a \leq \bigvee F$ [$a \leq b$]). So problems concerning (i) the existence of a maximal or the largest proper subalgebra of A including a given proper subalgebra B, and (ii) the existence of a maximal or the largest subalgebra of A incomparable with a given B in Sa(A), possess a lattice-theoretic character.

In this paper we study problems like (i) and (ii) in a few kinds of algebraic lattices. The present introductory section is followed by five other sections.

In Section 2 we deal with the "folklore" of the theory, i.e. with basic definitions and results on algebraic lattices, the class of subalgebras Sa(A) of a given algebra A, and their connection. In that section we also establish some straightforward results on lattice-theoretic formulations of the problems (i) and (ii) (Theorems 2.1 - 2.6) and their interpretation in the algebraic lattice Sa(A) (Theorems 2.9 and 2.10).

Sections 3 and 4 constitute the core of the present paper. In Section 3 we define GG-lattices [GH-lattices] , i.e. [strongly] algebraic lattices satisfying two additional axioms (cf. Definition 3.1 for details). These lattices turn out to be uncountable and each of their elements (apart from the zero and unit) possesses an incomparable element (Theorem 3.3). Our main results on GG- and GH-lattices (Theorems 3.5 and 3.7) characterize those

elements having a largest incomparable element: (1) x has a largest incomparable element iff x is strongly compact and unequal to the zero, and (2) x' is the largest element incomparable with x iff x is the smallest element incomparable with x'.

In Section 4 we introduce two special kinds of algebras, called GG- and GH-algebra, and we show that for each such algebra A, the lattice Sa(A) is a GG- or GH-lattice respectively. The last two sections contain applications to Abstract Families of Languages or AFL (Section 5) and some concluding remarks (Section 6).

This paper originates from an attempt to generalize results on incomparable AFL's due to Ginsburg and Spanier [18] to other AFL-structures (cf. [2] for a survey). Instead of repeating slightly modified proofs from [18] for each particular AFL-structure, we followed the uniform and algebraic approach sketched above. At appropriate places we indicate which of our propositions may be considered as lattice-theoretic or universally algebraic generalizations of theorems obtained by Ginsburg and Spanier [18]. Apart from their own algebraic interest the main results of this paper also show how little of the AFL-properties one actually needs (cf. Definition 4.2) in order to establish Ginsburg and Spanier-like theorems on AFL's and related algebraic structures.

2. Algebraic Lattices and Universal Algebra

In this section we recall some basic definitions in order to fix notation and terminology. Then we quote a few standard results from literature concerning the algebraic lattice of subalgebras of a given algebra. We also establish some elementary facts on incomparable elements in algebraic lattices.

The reader is assumed to be familiar with set theory, e.g. with the content of [25]. Recall that an element s of a nonempty partially ordered set $(S,\leq)$ is maximal, if for each t in S, $s \leq t$ implies $s = t$. An element s of $(S,\leq)$ is called a largest element of S if $t \leq s$ for all t in S. Clearly, a partially ordered set may have several maximal elements but at most one largest element. At a few places in this paper we need the Axiom of Choice. From its various equivalent formulations we use

Zorn's Lemma. If each chain in a nonempty partially ordered set has an upper bound, then this set has a maximal element. □

With respect to lattice theory and universal algebra this paper is almost self-contained. For more material on lattice theory and universal algebra the interested reader is referred to [5, 6, 9, 12, 21] and [6, 7, 19, 20, 26, 27, 30] respectively.

A binary operation $\cdot$ on a set S is called associative [commutative, idempotent] if for all $a,b,c \in S$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ [$a \cdot b = b \cdot a$, $a \cdot a = a$ respectively].

A lattice L is a nonempty set provided with two binary operations $\vee$ (called join or lub, i.e. least upper bound) and $\wedge$ (called meet or glb, i.e. greatest lower bound) which are both associative, commutative, idempotent and which are coupled by the following two "absorption laws" $a \wedge (a \vee b) = a$ and $a \vee (a \wedge b) = a$ for all $a,b \in L$. A lattice L is partially ordered by $\leq$, where for all $a,b \in L$, $a \leq b$ whenever $a \wedge b = a$ or, equivalently, whenever $a \vee b = b$. As usual we write $a < b$ if $a \leq b$ and $a \neq b$. If a lattice L has a smallest [largest] element with respect to $\leq$, then this unique element, denoted by $\perp$ [$\top$] is called the zero [unit] of L. Clearly, we have for each a in L, $a \wedge \perp = \perp$, $a \wedge \top = a$, $a \vee \perp = a$ and $a \vee \top = \top$.

Let S be an arbitrary subset of a lattice L. We say that an element a of L is an upper bound [a lower bound] of S, when $s \leq a$ [$a \leq s$] for each s in S. An element a of L is called the join or lub [meet or glb] of S, denoted by $a = \bigvee S$ [$a = \bigwedge S$] if a is an upper bound [a lower bound] of S and $a \leq b$ [$b \leq a$] for every upper bound [lower bound] b of S. We have $\bigvee\{a,b\} = a \vee b$ and $\bigwedge\{a,b\} = a \wedge b$. A lattice L is complete when for each subset S of L, $\bigvee S$ and $\bigwedge S$ exist. Each complete lattice L possesses a zero and a unit defined by $\perp = \bigwedge L$ and $\top = \bigvee L$ respectively.

An element a of a lattice L is called [strongly] compact if, whenever $a \leq \bigvee S$ for some nonempty subset S of L, there exists a finite subset F of S [an element b of S] with $a \leq \bigvee F$ [$a \leq b$]. An element a of L is called coprime if, whenever $a \leq \bigvee F$ for some nonempty finite subset F of L, there exists an element b in F with $a \leq b$. Obviously, $a \in L$ is strongly compact iff a is both compact and coprime. Note that the zero $\perp$ in a lattice is strongly compact. A lattice L is said to be [strongly] algebraic or [strongly] compactly generated if L is complete and each element in L is the join of some set of [strongly] compact elements of L.

Two elements a and b of partially ordered set are called [in]comparable, denoted by a:b [a%b], if either $a \leq b$ or $b \leq a$ [neither $a \leq b$ nor $b \leq a$]. Note that a:b iff b:a, and a%b iff b%a.

The existence of maximal [largest] incomparable elements is rather simple to establish for those [strongly] compact elements, which already posses an incomparable element, since we have (cf. [18] Theorem 3.1)

Theorem 2.1. Let c and d be incomparable elements in a complete lattice L.

(1) If c is compact, then there exists in L a maximal element incomparable with c and $\geq d$.

(2) If c is strongly compact, then there exists in L a largest element incomparable with c.

Proof. (1) We apply Zorn's Lemma to the nonempty partially ordered set $P = \{a \in L \mid a \% c;\ d \leq a\}$. Let A be a chain in P. We show that $\bigvee A \in P$.

Clearly $d \leq \bigvee A$. Suppose $c : \bigvee A$. If $\bigvee A \leq c$, then $d \leq c$ contradicting $c \% d$. Therefore $c \leq \bigvee A$. Since c is compact, there exists a finite subchain F for A with $c \leq \bigvee F$. Let f be the largest element of F. Then $c \leq f$; but this is impossible as f is in A. Consequently $c \% \bigvee A$, and hence $\bigvee A \in P$. By Zorn's Lemma there exists a maximal element incomparable with c and $\geq d$.
(2) Let $B = \{b \in L \mid b \% c\}$. Obviously, B is partially ordered and nonempty, because $d \in B$. Suppose $c : \bigvee B$. If $\bigvee B \leq c$, then $d \leq c$ contradicting $c \% d$. Therefore $c \leq \bigvee B$, Since c is strongly compact there exists an element $b_0 \in B$ with $c \leq b_0$. But this contradicts $b_0 \% c$. Hence $c \% \bigvee B$ and consequently $\bigvee B \in B$. □

An element e in a lattice L splits [24] if there exists $a, b \in L$ such that $a \% b$ and $e = a \vee b$. Clearly, a coprime element does not split. In general the converse is however not true; cf. [18] p. 97 and Proposition 2.5 below.

We now turn to maximal or largest elements properly smaller than a given element. These elements also play an important part in the study of largest incomparable elements (cf. 3.4 and 3.5 below).

Theorem 2.2. Let b and c be elements in a complete lattice L, such that $b < c$.

(1) If c is compact, then there exists a maximal element $c_b \in L$ such that $b \leq c_b < c$.

(2) If c is compact and does not split, then there exists a largest element $c_\ell \in L$ such that $c_\ell < c$.

Proof. (1) We apply Zorn's Lemma to the nonempty and partially ordered set $P = \{x \in L \mid b \leq x < c\}$. Let A be a chain in P. Certainly we have $b \leq \bigvee A \leq c$. If $c = \bigvee A$, then the compactness of c implies that there is a finite subchain F of A such that $c \leq \bigvee F$. Let f be the largest element of F, then $c \leq f$ contrary to the fact that $f \in P$. Thus $\bigvee A < c$ and $\bigvee A \in P$.

From Zorn's Lemma we conclude that P contains a maximal element c_b with $b \leq c_b < b$.

(2) Let $B = \{x \in L \mid x < c\}$. Then B is a nonempty partially ordered set and $\bigvee B \leq c$. If $c = \bigvee B$, then the compactness of c implies the existence of a finite set $F \subseteq B$ with $c \leq \bigvee F \leq \bigvee B = c$, i.e. $c = \bigvee F$. Suppose that F has a largest element f. Then $c = f \in B$ contradicting the fact $c \notin B$. Therefore F has a finite number of maximal elements $f_1, \ldots, f_n$ $(n \geq 2)$ such that $c = f_1 \vee \ldots \vee f_n$. But this easily implies that c splits. Hence $\bigvee B < c$.

From the definition of B it follows that $\bigvee B = c_\ell$. □

Corollary 2.3. For each strongly compact element c in a complete lattice L with $c \neq 1$, there exists a largest element $c_\ell \in L$ such that $c_\ell < c$. □

For algebraic lattices we have the following converse of 2.2(2).

Theorem 2.4. Let a be an element of an algebraic lattice L. If there exists a largest element $a_\ell \in L$ such that $a_\ell < a$, then a is compact and does not split.

Proof. Consider $Q = \{x \in L \mid x < a\}$ and $C = \{c \in L \mid c \leq a, c \text{ is compact}\}$. Clearly $a_\ell = \bigvee Q$, and since L is algebraic we have $a = \bigvee C$ and $a_\ell = \bigvee (C \cap Q)$. As $a_\ell < a$, we have $C \cap Q \subset C$. Therefore there exists an element in C−Q which has to be a. Hence a is compact.

Suppose a splits, i.e. $a = d \vee e$ and $d \not\leq e$, then $d < a$, $e < a$ and $d \vee e \leq a_\ell < a = d \vee e$. Hence a does not split. □

For a special class of algebraic lattices a converse of 2.1(2) will be established in Section 3 (cf. Theorem 3.5).

A lattice L is called distributive if $(a \wedge b) \vee (a \wedge c) = a \wedge (b \vee c)$ or equivalently, if $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ for all $a,b,c \in L$. A complete lattice L is called completely distributive if, for every doubly indexed family $a_{i,j}$ $(i \in I, j \in J)$ in L we have $\bigwedge\{\bigvee\{a_{i,j} \mid j \in J\} \mid i \in I\} = \bigvee\{\bigwedge\{a_{i,\phi(i)} \mid i \in I\} \mid \phi \in J^I\}$ where J^I denotes the set of all functions on I to J.

In distributive lattices the notions of coprime and non-splitting element coincide. The proof is straightforward (cf. [6] pp. 11-12, [9] pp. 19-21) and it is therefore omitted.

Proposition 2.5. The following statements are equivalent for each lattice L.

(i) L is distributive.

(ii) An element in L is coprime if (and only if) it does not split. □

For algebraic lattices we have the following result characterizing complete distributivity (cf. [4, 12]).

Theorem 2.6. An algebraic lattice is completely distributive if and only if it is strongly algebraic.

Proof. The "if" part has been established by Raney [28] (for arbitrary complete lattices; cf. [3], [5] p. 245).

In order to prove the converse implication, it suffices to show that each compact element c in a completely distributive algebraic lattice L

is the join of strongly compact elements.

If $c = \perp$, then c is strongly compact. So we assume $\perp < c$.

Let C be the set of all maximal elements m in L such that $\perp \leq m < c$. By 2.2(1) C is nonempty. If C is a singleton, then c possesses a largest element c_ℓ such that $c_\ell < c$. From 2.4 and 2.5 it follows that c is strongly compact.

Assume now that $C = \{m_i \mid i \in I\}$ is not a singleton. For each $i \in I$ define $M_i = \{x \in L \mid x < c, x \not\leq m_i\}$. Since C contains at least two elements, each M_i is nonempty.

Suppose $\bigwedge M_i \leq m_i$. Then $m_i = m_i \vee \bigwedge M_i = \bigwedge\{m_i \vee x \mid x < c, x \not\leq m_i\}$. From the maximality of m_i, $m_i \leq m_i \vee x \leq c$ and $x \not\leq m_i$ $(x \in M_i)$, we obtain $m_i \vee x = c$, and hence $m_i = \bigwedge c = c$, contradicting $m_i < c$. Therefore $\bigwedge M_i \not\leq m_i$, and hence $m_i \wedge \bigwedge M_i < \bigwedge M_i$.

Let y be an element with $y < \bigwedge M_i$. Then $y \notin M_i$ and so $y \leq m_i$. Hence $y \leq m_i \wedge \bigwedge M_i$. But this means that $m_i \wedge \bigwedge M_i$ is the largest element of $\{y \in L \mid y < \bigwedge M_i\}$. By 2.4 and 2.5, it follows that $\bigwedge M_i$ is strongly compact.

Since $\bigwedge M_i \not\leq m_i$ $(i \in I)$, we have $\bigvee\{\bigwedge M_i \mid i \in I\} \not\leq m_j$ for each $j \in I$. Hence $c = \bigvee\{\bigwedge M_i \mid i \in I\}$, i.e., c is the join of strongly compact elements. □

Let L be a lattice. A function $\Gamma : L \to L$ is a closure operator on L if, for all a and b in L

(i) $a \leq \Gamma(a)$ (Γ is extensive)

(ii) $a \leq b$ implies $\Gamma(a) \leq \Gamma(b)$ (Γ is monotone)

(iii) $\Gamma(\Gamma(a)) = \Gamma(a)$ (Γ is idempotent)

A [completely] additive closure operator Γ on L is a closure operator satisfying $\Gamma(\bigvee S) = \bigvee\{\Gamma(s) \mid s \in S\}$ for each finite subset [arbitrary subset] S of L.

If L is a [strongly] algebraic lattice, then a closure operator Γ on L is called [strongly] algebraic if for each [strongly] compact element c in L with $c \leq \Gamma(b)$ for some b in L, there exists a [strongly] compact element $b' \leq b$ with $c \leq \Gamma(b')$.

Theorem 2.7. (1) If Γ is a closure operator on a complete lattice L, then $\Gamma(L) = \{\Gamma(a) \mid a \in L\}$ is a complete lattice in which the meet coincides with the meet in L, whereas the join $\bigvee'$ in $\Gamma(L)$ satisfies $\bigvee' S = \Gamma(\bigvee S)$ for each $S \subseteq \Gamma(L)$, where $\bigvee$ is the join in L.
(2) If Γ is a [completely] additive closure operator on a [completely] distributive complete lattice L, then $\Gamma(L)$ is [completely] distributive.
(3) If Γ is a [strongly] algebraic closure operator on a [strongly] algebraic lattice L, then $\Gamma(L)$ is a [strongly] algebraic lattice.

Proof. (1) is well known (cf. [6] pp. 112, or [5] Theorem II.4.12).
(2) Let $a_{i,j}$ $(i \in I, j \in J)$ be in L, where I and J are finite [arbitrary] index sets. By (1), the [complete] distributivity and the fact that Γ is [completely] additive, we have $\bigwedge\{\bigvee'\{a_{i,j} \mid j \in J\} \mid i \in I\} = \bigwedge\{\bigvee\Gamma(\{a_{i,j} \mid j \in J\}) \mid i \in I\} = \bigwedge\{\bigvee\{\Gamma(a_{i,j}) \mid j \in J\} \mid i \in I\} = \bigvee\{\bigwedge\{\Gamma(a_{i,\phi(i)}) \mid i \in I\} \mid \phi \in J^I\} = \bigvee\{\Gamma(\bigwedge\{a_{i,\phi(i)} \mid i \in I\}) \mid \phi \in J^I\} = \Gamma(\bigvee\{\bigwedge\{a_{i,\phi(i)} \mid i \in I \mid \phi \in J^I\}) = \bigvee'\{\bigwedge\{a_{i,\phi(i)} \mid i \in I\} \mid \phi \in J^I\}$, i.e. $\Gamma(L)$ is [completely] distributive.
(3) The algebraic case is standard ([6] VII, [26], [30] § 4 Lemma 2) whereas the proof of the strongly algebraic variant can be obtained as a simple modification. □

We now show how these lattice-theoretic concepts can be used in universal algebra. An algebra (A,Ω) is a pair consisting of a set A

(called the carrier set) and a set of operations $\Omega = \{f_\xi \mid 0 \leq \xi < \alpha\}$ (α is an arbitrary ordinal number), such that A is closed under each f_ξ from Ω. Let $\sigma(f_\xi)$ denote the arity or rank of the operation f_ξ. We call an algebra (A,Ω) [pseudo]unary if $\sigma(f_\xi) = 1$ [$\sigma(f_\xi) \leq 1$] for each $f_\xi \in \Omega$, i.e. (A,Ω) only possesses unary operations [and distinguished elements]. In contravention of standard texts (cf. [5, 6, 7, 20, 26, 27, 30]) we allow the carrier set A to be empty provided that $\sigma(f_\xi) \neq 0$ for each f_ξ in Ω (cf. [19]).

An algebra (B,Ω') is a subalgebra of the algebra (A,Ω) if (i) $B \subseteq A$, (ii) Ω' consists of the restrictions g_ξ of f_ξ from Ω with respect to B, i.e. $g_\xi = f_\xi \mid B$ $(0 \leq \xi < \alpha)$, and (iii) B is closed under each g_ξ from Ω'.

Whenever Ω (and consequently each Ω') is understood, we will identify algebras with their carrier set. Thus we use the same symbol A to denote both the algebra (A,Ω) and its carrier set A; from the context it should always be clear which is meant.

Let A be an algebra. We denote the class of all subalgebras of A by Sa(A). For $B_1, B_2 \in Sa(A)$ we write $B_1 \leq B_2$ whenever B_1 is a subalgebra of B_2 and so Sa(A) is obviously provided with a partial order.

With the algebra A we can associate a closure operator Γ mapping each subset $X \subseteq A$ into the carrier set of the smallest subalgebra B of A such that $X \subseteq \Gamma(X) = B$. We also say that B is the subalgebra generated by X. The algebra B is said to be finitely generated [cyclic] if there exists a finite set [singleton] X such that $\Gamma(X) = B$.

Let A be an arbitrary but fixed algebra and consider the power set of the carrier set A. It is well known that this power set (just as each other power set) is a completely distributive complete lattice with the set-theoretic inclusion as partial order, $\perp = \emptyset$ and $\top = A$. It is easy to verify that the [strongly] compact elements of this lattice are exactly the finite

sets [singletons together with $\emptyset$] and that this lattice is strongly algebraic.

In view of Theorem 2.7 the major part of the following basic result should not be very surprising.

Theorem 2.8. (1) For each algebra A, the set Sa(A) of all subalgebras of A is an algebraic lattice with for each $S \subseteq Sa(A)$, $\bigwedge S = \bigcap S$, $\bigvee S = \Gamma(\bigcup S)$, $\top = A$ and $\bot = \Gamma(\emptyset)$, where Γ is the closure operator associated with A. Moreover

(i) Γ is an algebraic closure operator satisfying

$$\Gamma(X) = \bigcup\{\Gamma(X_f) \mid X_f \subseteq X, X_f \text{ is finite}\}$$

(ii) An element B in Sa(A) is compact if and only if B is finitely generated.

(2) If A is a pseudo-unary algebra, then $\bigvee S = \bigcup S$ for each $S \subseteq Sa(A)$, and

(iii) Γ is a completely additive strongly algebraic closure operator.

(iv) Sa(A) is a strongly algebraic lattice.

Proof. The main part is (up to a reformulation) equal to standard results proved in [6] VIII.5-4, [7] II.5, [20]6 & 9 and [26] 3.6 & 3.8. The only new proposition is (iv) which directly follows from the complete distributivity of Sa(A) and Theorem 2.6. □

A statement similar to 2.8(1)(ii) for strongly compact elements and cyclic subalgebras respectively, does not hold in general (cf. e.g. 5.2 and 5.3 below).

By Theorem 2.8 terminology and results on (algebraic) lattices become applicable to Sa(A). So phrases like incomparable (sub)algebra, non-splitting algebra, coprime algebra, etc. are meaningful. From 2.1 and 2.8 we obtain

Theorem 2.9. Let C and D be incomparable subalgebras of an algebra A.

(1) If C is finitely generated, then there exists a maximal subalgebra of A, including D and incomparable with C.

(2) If C is finitely generated and coprime, then there exists a largest subalgebra of A incomparable with C. □

An element a of the carrier set A of an algebra is said to be a non-generator of the algebra, if for each set $X \subseteq A$, the condition that $X \cup \{a\}$ generates A, implies that X generates A. Let N(A) denote the set of all non-generators of A.

Theorem 2.10. Let C be an algebra.

(1) If C is finitely generated, then each proper subalgebra of C is included in a maximal proper subalgebra of C.

(2) N(C) is the intersection of all maximal proper subalgebras of C. Consequently, if C has a proper subalgebra, then N(C) is a proper subalgebra of C.

(3) Let C have a proper subalgebra. Then the following are equivalent.

(i) C is finitely generated and does not split.

(ii) N(C) is the largest proper subalgebra of C.

Proof. (1) directly follows from 2.2(1) and 2.8 (cf. [26] pp. 80-83), whereas (2) has been established in [26] pp. 80-83. Finally, (3) is obtained from (2), 2.2(2), 2.4 and 2.8. □

3. GG- and GH-lattices

In order to characterize largest incomparable elements in algebraic lattices we introduce in this section two particular kinds of algebraic

lattices (Definition 3.1; [18] Lemma 1.1) for which we establish our main results (Theorems 3.5 and 3.7). But first, we consider the following obvious generalization of the notion of compactness (cf. [19]).

An element a in a lattice L is called weakly compact if, whenever $a \leq \bigvee S$ for some nonempty $S \subseteq L$, there exists a countable subset S_0 of S with $a \leq \bigvee S_0$. Clearly, each compact element is weakly compact.

Definition 3.1. A Ginsburg lattice or GG-lattice is an algebraic lattice L satisfying the following properties

(G1) For each weakly compact element a in L, $x \leq a$ implies that x is weakly compact.

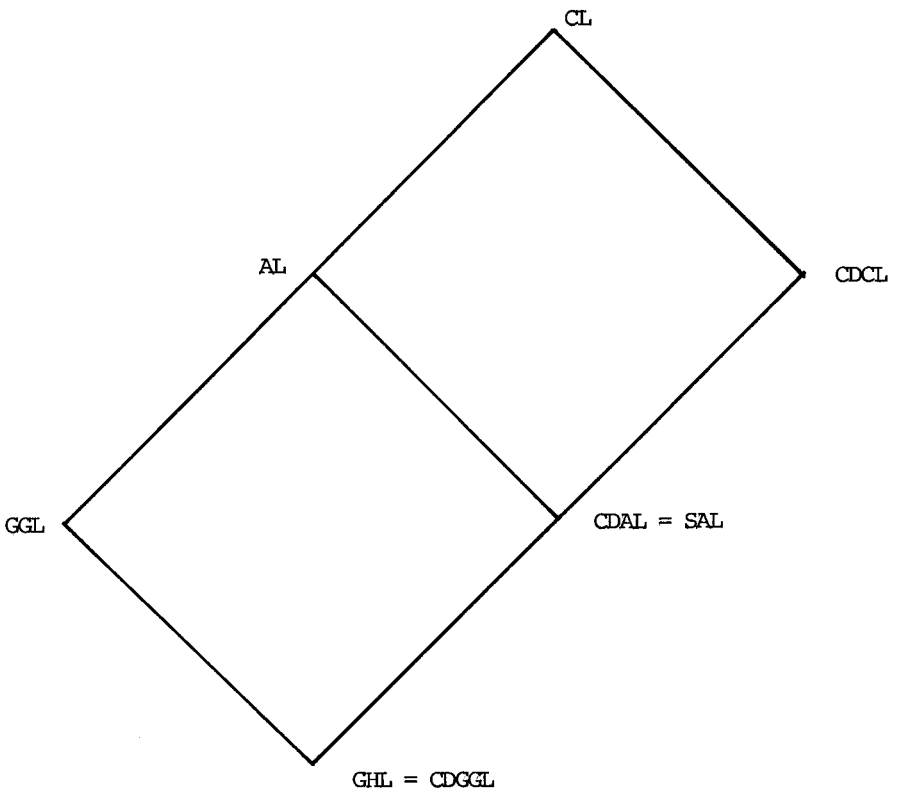
(G2) For each pair of weakly compact elements a and b in L there exists a weakly compact element c in L such that $a < c$ and $c \wedge b = a \wedge b$.

A strongly algebraic GG-lattice is called a Greibach or GH-lattice. □

Property (G2) says that for each pair of weakly compact elements a and b, a can be properly enlarged (to c) without changing its meet with b. Thus in a GG-lattice the unit is not weakly compact. In Section 4 we show that in the algebraic lattice Sa(A) of subalgebras of (A,Ω) property (G1) corresponds to the fact that Ω consists of a countable number of operations.

The main types of lattices introduced in Sections 2 and 3 are surveyed in Figure 1.

Lemma 3.2. (cf. [18] Lemma 1.2). Let x with $x \neq \top$ be an element of a GG-lattice L. If b is weakly compact in L with $x \not\leq b$, then there exists a weakly compact element y in L such that $b < y$ and $x \% y$.



A = algebraic; C = complete(ly); D = distributive;

L = lattices; S = strongly; GG- and GH-lattices are

defined in Section 3.

Figure 1.

Proof. Since $\top$ is not weakly compact, we have $b \neq \top$.

Distinguish the following cases.

Case 1: x is not weakly compact.

Since L is algebraic we have for each e in L, $e = \bigvee\{c \mid c \text{ is compact}; c \leq e\}$. Hence $b \neq \top$ and $x \neq \top$ imply the existence of compact elements d and z, such that $d \not\leq b$ and $z \not\leq x$. Define y by $y = b \vee d \vee z$. Being the join of weakly compact elements, y is also weakly compact. Furthermore we have $b < y$ (Otherwise $d \leq b$ holds, contradicting $d \not\leq b$). Since $z \leq y$ and $z \not\leq x$, we have $y \not\leq x$. On the other hand, (G1) yields $x \not\leq y$. Hence $x \% y$.

Case 2: x is weakly compact.

By (G2) there exists a weakly compact element y with $b < y$ and $y \wedge x = b \wedge x$. Suppose $y \leq x$. Then $y = y \wedge x = b \wedge x$. Since $b \leq y \leq x$ we have $b \wedge x = b$ and, consequently $y = b$, which contradicts $b < y$. Suppose $x \leq y$. Then $x = y \wedge x = b \wedge x$ or, equivalently, $x \leq b$, contradicting $x \not\leq b$. Hence $x \% y$. □

Theorem 3.3. (cf. [18] Theorem 1.1).

(1) Each GG-lattice is uncountable.

(2) Each element x in a GG-lattice satisfying $\bot < x < \top$ possesses a weakly compact element incomparable with x.

Proof. (1) Let L be a GG-lattice with unit $\top$. Since $\top$ is not weakly compact, and $\top = \bigvee L$, L is uncountable.

(2) Lemma 3.2 with $b = \bot$. □

We are now almost able to establish the first characterization result (Theorem 3.5) for which we need the following interesting lemma and the notion of relative pseudo-complement.

Lemma 3.4. (cf. [18] Proposition 2.1 and Corollary). Let L be a GG-lattice and $x \in L$.

(1) If x has a largest incomparable element x', then $x \wedge x'$ is the largest element in L strictly smaller than x.

(2) If x has a largest incomparable element, then x is compact and does not split.

Proof. (1) From $x \% x'$ it follows that $x \wedge x' \neq x$ and hence $x \wedge x' < x$. Let z be in L such that $z < x$. In order to prove that $z \leq x \wedge x'$ it suffices to show that $c \leq x \wedge x'$ for each compact element c with $c \leq z$, because L is algebraic (i.e. each element in L is the join of compact elements). Obviously, c is weakly compact, $x \neq \top$ and $x \not\geq c$. Thus by 3.2 there exists an element y in L with $c < y$ such that $x \% y$. This implies $c < y \leq x'$, since x' is the largest element incomparable with x. Together with $c \leq z \leq x$, this yields $c \leq x \wedge x'$.

(2) directly follows from (1) and 2.4. □

Let a and b be elements in a lattice L. If there exists a largest element x in L such that $a \wedge x \leq b$, then this element is denoted by $a \circ b$ and is called the relative pseudo-complement of a with respect to b.

Theorem 3.5. (cf. [18] Theorem 2.1). Let x be an element of a GG-lattice L. Then the following are equivalent.

(i) x is strongly compact and $\perp < x$.

(ii) There exists a largest element x' in L such that $x \% x'$.

(iii) There exists a largest element x_ℓ in L such that $x_\ell < x$, whereas $x \circ x_\ell$ exists and $x \circ x_\ell$ is (the largest element) incomparable with x.

Proof. (i) $\Longrightarrow$ (ii). By Theorems 2.1(2), 3.3(2) and the fact that $\top$ is not strongly compact.

(ii) $\Longrightarrow$ (iii). According to 3.4(1) we have for the largest element x_ℓ satisfying $x_\ell < x$, $x \wedge x' = x_\ell$. Hence $x \circ x_\ell$ exists and $x' \leq x \circ x_\ell$.

Let z satisfy $x \wedge z \leq x_\ell$. Suppose $x \leq z$. Then $x \wedge z = x < x_\ell$, which is impossible. Therefore, either $z \leq x_\ell < x$ or $z \% x$, which in both cases yields $z \leq x'$. Hence $x \circ x_\ell = \bigvee\{z \mid x \wedge z \leq x_\ell\} \leq x'$.

(iii) $\Longrightarrow$ (i). Since there is an element incomparable with x, we have $\bot < x < \top$. By 2.4, x is compact. So it remains to prove that x is coprime.

Suppose x is not coprime, i.e. there exists a finite set F such that $x \leq \bigvee F$ and $x \not\leq f$ for each $f \in F$. Then $x \wedge f \leq x_\ell$ and so $f \leq x \circ x_\ell$ for each f in F. Consequently, $x \leq \bigvee F \leq x \circ x_\ell$, contradicting $(x \circ x_\ell) \% x$. Hence x is coprime. □

Besides this result we give another characterization of the largest incomparable element in a GG-lattice (Theorem 3.7) for which we need the following

Lemma 3.6. (cf. [18] Lemma 2.2) If an element x' of a GG-lattice has a smallest incomparable element x, then (i) x is compact, and (ii) x' is not weakly compact.

Proof. Since L is algebraic, there exists a compact element c such that $c \leq x$ and $c \not\leq x'$. But then $c \% x'$ (Otherwise we have $x' \leq c \leq x$, contradicting $x \% x'$). Hence $x \leq c$, because x is the smallest element incomparable with x'. But this means $x = c$, i.e. x is compact.

Suppose x' is weakly compact. Then $x \vee x'$ and by (G1) $x \wedge x'$ are also weakly compact, and $x \vee x' < \top$. Applying (G2) to $x \wedge x'$ and $x \vee x'$

yields the existence of a weakly compact element w, such that $x \wedge x' < w$ and $w \wedge (x \vee x') = (x \wedge x') \wedge (x \vee x') = x \wedge x'$.

Consider w and x'. First, $w \leq x'$ is impossible, as this implies $w \leq x \vee x'$ and hence $w = w \wedge (x \vee x') = x \wedge x'$, contradicting the fact that $x \wedge x' < w$. On the other hand the supposition $x' \leq w$ implies $x' \leq w \wedge (x \vee x') = x \wedge x'$ or, equivalently $x' \leq x$, contradicting $x \% x'$. Hence $w \% x'$.

Since x is the smallest element incomparable with x', we have $x \leq w$. But this implies $x \leq w \wedge (x \vee x') = x \wedge x'$, or $x \leq x'$, contradicting $x \% x'$.

Hence x' is not weakly compact. □

Theorem 3.7. (cf. [18] Theorem 2.2). Let x and x' be elements in a GG-lattice L. Then the following propositions are equivalent.

(i) x' is the largest element in L incomparable with x.

(ii) x is the smallest element in L incomparable with x'.

Proof. (i) $\Longrightarrow$ (ii). Let y be incomparable with x'. We show that $x \leq y$.

Suppose $x \% y$. Then $y \leq x'$ because x' is the largest element incomparable with x. This contradicts $y \% x'$. Suppose $y < x$. From 3.4(1) we obtain $y \leq x_\ell = x \wedge x' \leq x'$, contradicting $y \% x'$. Hence we have $x \leq y$.

(ii) $\Longrightarrow$ (i). Define $E(x) = \{c \mid c \text{ is compact; } x \not\leq c\}$. Since L is algebraic we have for each element y incomparable with x, $y \leq \bigvee E(x)$. In particular $x' \leq \bigvee E(x)$.

We show that $\bigvee E(x) \leq x'$. Let c be an element from E(x). Then $x \not\leq c$. Since x is the smallest element incomparable with x', we have $c : x'$. By 3.6 x' is not weakly compact and according to (G1) this implies $x' \not\leq c$. Hence $c \leq x'$, and consequently $\bigvee E(x) \leq x'$. By $x \% x'$ and the definition of E(x), it follows that x' is the largest element incomparable with x. □

4. GG- and GH-algebras

In this section we introduce two particular kinds of algebras called GG- and GH-algebras (Definition 4.2). The main result of this section (Theorem 4.3) says that for each GG-algebra [GH-algebra] A the lattice of subalgebras Sa(A) is a GG-lattice [GH-lattice] . First, we establish however a preliminary result and we recall some auxiliary concepts.

Proposition 4.1. Let A be an algebra with a countable set of operations Ω.

(i) A subalgebra B of A is weakly compact in Sa(A) if and only if B has a countable carrier set.

(ii) Sa(A) satisfies property (G1).

Proof. Since Sa(A) is algebraic we have $B = \bigvee\{C_\lambda \mid \lambda \in \Lambda\}$, where each C_λ is the carrier set of a finitely generated algebra (Theorem 2.8). If B is weakly compact, then there exists a countable subset Λ_ω of Λ such that $B = \bigvee\{C_\lambda \mid \lambda \in \Lambda_\omega\}$. By 2.8(1) we have $B = \Gamma(\bigcup\{C_\lambda \mid \lambda \in \Lambda_\omega\})$ where Γ is the closure operator associated with A. As Ω is countable, each C_λ and consequently $\bigvee\{C_\lambda \mid \lambda \in \Lambda_\omega\}$ is countable. According to [20] p. 47 or [26] p. 78 the algebra B has a countable carrier set.

Conversely, suppose that B has a countable carrier set. Let $B \subseteq \bigvee\{D_\lambda \mid \lambda \in \Lambda\}$ where each D_λ is a subalgebra of A. Since Ω is countable, for each element b in B there exists a countable subset Λ_b of Λ such that $b \in \Gamma(\bigcup\{D_\lambda \mid \lambda \in \Lambda_b\})$. Define $\Lambda_\omega = \bigcup\{\Lambda_b \mid b \in B\}$ which is obviously countable. Moreover $B \subseteq \Gamma(\bigcup\{D_\lambda \mid \lambda \in \Lambda_\omega\}) = \bigvee\{D_\lambda \mid \lambda \in \Lambda_\omega\}$, i.e. B is weakly compact.

(ii) follows from (i) and the fact that each countable algebra can only posses countable subalgebras. □

Let f be an n-ary operation on a collection of sets. Then f is called compact if $f(S_1,\ldots,S_n) = \bigcup\{f(S_1',\ldots,S_n') \mid S_i' \subseteq S_i;\ S_i' \text{ is finite};\ 1 \leq i \leq n\}$, and monotone if $S_i \subseteq T_i$ for each i $(1 \leq i \leq n)$ implies $f(S_1,\ldots,S_n) \subseteq f(T_1,\ldots,T_n)$.

Definition 4.2. A Ginsburg algebra or GG-algebra is an algebra (A,Ω) satisfying the following conditions.

(i) The carrier set A is a collection of sets (called objects).

(ii) Ω is a countable set of operations which are all both monotone and compact.

(iii) Ω contains a 0-ary operation f_0 such that $f_0 = S_0$, $S_0 \in A$ and S_0 is a countably infinite set (S_0 will be called the central object of A).

(iv) Each subset of the central object S_0 is in A.

A Greibach or GH-algebra is a pseudo-unary GG-algebra.[1] □

Theorem 4.3. If A is a GG-algebra [GH-algebra], then Sa(A) is a GG-lattice [GH-lattice].

Proof. According to 2.8., 3.1, 4.1 and 4.2 it only remains to show that for each GG-algebra A, the lattice Sa(A) satisfies (G2). The argument is a straightforward generalization of a proof technique due to Ginsburg and Spanier ([18] Lemma 1.1; cf. [14] pp. 208-211).

By 2.8 and 4.1 it suffices to prove that for each pair of countable proper subalgebras B_1 and B_2 of A, there exists an object S not in the carrier set of B_2 such that $\Gamma(B_2 \cup \{S\}) \cap B_1 = B_2 \cap B_1$, where Γ is the closure operator associated with A.

Let Ψ be the set of all operations from Ω augmented with the identity

operation and all unary constant operations $C_R(X) = R$ for each object R in B_2. Since both B_2 and Ω are countable, so is Ψ.

Define Φ_0 to be the set of all operations ϕ defined by $\phi(X) = \psi(X,\ldots,X)$, where ψ is an m-ary operation from Ψ. For each $n \geq 0$, define Φ_{n+1} to be the set of all operations $\phi(X) = \psi(\phi_1(X),\ldots,\phi_m(X))$, where ψ is an m-ary operation from Ψ and $\phi_1,\ldots,\phi_m \in \Phi_n$. Finally, let $\Phi = \bigcup\{\Phi_n \mid n \geq 0\}$. As the composition of monotone [compact] operations is monotone [compact], each operation in Φ is both monotone and compact.

Let $\{S_i\}_{i\geq 1}$, $\{\phi_i\}_{i\geq 1}$ and $\{Q_i\}_{i\geq 1}$ be respectively enumerations of the objects in $B_1 - B_2$, the elements of Φ, and the infinite objects in B_2. We define an object S satisfying, (α) S is an infinite subset of the central object S_0 of B_2, (β) $S \neq Q_k$ for each $k \geq 1$, and (γ) for each pair $(i,j) \in N \times N$, $\phi_i(S) \neq S_j$, where N is the set of nonnegative numbers. From (β) it follows that $S \notin B_2$, and by (γ) we obtain $\Gamma(B_2 \cup \{S\}) \cap B_1 = \{\phi(S) \mid \phi \in \Phi\} \cap B_1 = B_2 \cap B_1$.

Consider a bijection $\pi : N \times N \to N$. Let $U_{-1} = V_{-1} = \emptyset$. Suppose that for each $t < k$ with $k \geq 0$, finite disjoint subsets U_t and V_t of S_0 have already been defined inductively, such that $U_i \subseteq U_{i+1}$ and $V_i \subseteq V_{i+1}$ for all i $(0 \leq i < k-1)$. The finiteness of both U_{k-1} and V_{k-1} implies the existence of an element u_k in $S_0 - (U_{k-1} \cup V_{k-1})$. Let i and j be the unique nonnegative integers satisfying $\pi(i,j) = k$. Let $U_k = U_{k-1} \cup \{u_k\}$ whenever $\phi_i(S_0 - V_{k-1}) \neq \emptyset$ and $\phi_i(S_0 - V_{k-1}) \subseteq S_j$. Suppose $\phi_i(S_0 - V_{k-1})$ is empty or not a subset of S_j. Since ϕ_i is compact, there exists a finite subset W_k of $S_0 - V_{k-1}$, such that $\phi_i(W_k)$ is either empty or not a subset of S_j. Define $U_k = U_{k-1} \cup \{u_k\} \cup W_k$. Since Q_k is infinite, in either case

there exists an element v_k in $Q_k - U_k$. Finally, let $V_k = V_{k-1} \cup \{v_k\}$, which extends the induction. Define $S = \bigcup\{u_n \mid n \geq 0\}$. By 4.2(iv) S is in A, i.e. $\Gamma(B_2 \cup \{S\})$ is a proper subalgebra of A.

The verification of (β) and (γ) — for which we need the compactness and the monotinicity of each ϕ in Φ — is similar as in [18] Lemma 1.1 (cf. [14] pp. 208-211. The objects are languages and a^* is the central object.) and it is therefore not repeated here. □

Following 4.3 results on incomparable elements in GG-lattices (Section 3) are now applicable to Sa(A) for each GG-algebra A.

If X is a subalgebra of A, then the exterior of X in Sa(A) [18] is defined by $E(X) = \bigcup\{C \mid C \in Sa(A);\ C \text{ is finitely generated};\ X \not\subseteq C\}$. It is straightforward to show that for each strongly compact subalgebra X of a GG-algebra A, we have $E(X) = X \circ X_\ell$ provided $X \neq \iota$ (cf. 3.5 and the proof of 3.7). Note that in an algebraic lattice we also have $N(X) = E(X) \cap X$.

We leave the formulation of 3.3, 3.5 and 3.7 in terms of GG-algebras and exterior to the reader.

5. Application to AFL-theory

In this section the reader is assumed to be familiar with the rudiments of Abstract Families of Languages (cf. [13] Chapters 1-3, 5, 6 or [29] Chapter IV).

Let Σ_ω be a countably infinite set of symbols. A family (of languages) is a collection of languages over finite subsets (called alphabets) of Σ_ω. A family is called nontrivial if it contains a nontrivial language L, i.e. L is nonempty and $L \neq \{\lambda\}$, where λ denotes the empty word.

Let K be a family. A [K-]substitution τ is a function on an alphabet such that for each $\sigma \in \Sigma$, $\tau(\sigma)$ is a language [in K]. The function τ is extended to words by $\tau(\lambda) = \{\lambda\}$, $\tau(\sigma_1 \ldots \sigma_n) = \tau(\sigma_1) \ldots \tau(\sigma_n)$ for each $\sigma_1 \ldots \sigma_n \in \Sigma^+$, and to languages by $\tau(L) = \bigcup\{\tau(w) \mid w \in L\}$ for each $L \subseteq \Sigma^*$ (note that $\tau(\emptyset) = \emptyset$). A [K-]substitution over Σ is a [K-]substitution τ with $\tau(\sigma) \subseteq \Sigma^*$ for each $\sigma \in \Sigma$. If K equals FIN, i.e. the family of finite languages [REG, i.e. the family of regular languages] then a K-substitution is usually called finite [regular] substitution.

For each $m,n \geq 1$ we introduce a few $(mn+1)$-ary operations. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ and let L_0, L_{ij} $(1 \leq i \leq m;\ 1 \leq j \leq n)$ be languages over Σ^*. An iterated substitution μ [31, 1] is an operation transforming $(L_0, L_{11}, \ldots, L_{mn})$ into the language $\bigcup\{f_1 \ldots f_k(L_0) \mid k \geq 0;\ f_p \in \{\tau_1, \ldots, \tau_m\}$ for $1 \leq p \leq k$; each τ_i is a substitution over Σ with $\tau_i(\sigma_j) = L_{ij}\}$ over Σ. If $m = 1$ [each L_{ij} contains σ_j] then μ is called a single [nested] iterated substitution. Similarly, each substitution on Σ is an $(n+1)$-ary operation.

Let ALL be the family of all languages over finite subsets of Σ_ω. Then the powerset of ALL, denoted by ALL, equals the class of all language families over Σ_ω (including the trivial families $\emptyset$, $\{\emptyset\}$, $\{\{\lambda\}\}$ and $\{\emptyset, \{\lambda\}\}$).

We now introduce some specific sets of operations Ω such that the pair (ALL,Ω) becomes an algebra. Define Ω_Π, Ω_{Π_0} and $\Omega_{\hat{\mathcal{RL}}}$ by $\Omega_\Pi = \{f_1\} \cup \{\delta_R \mid \delta_R(L) = L \cap R;\ R \in \mathrm{REG}\} \cup \{\phi \mid \phi$ is a finite substitution$\}$, $\Omega_{\Pi_0} = \Omega_\Pi \cup \{f_0\}$, and $\Omega_{\hat{\mathcal{RL}}} = \Omega_{\Pi_0} \cup \{\rho \mid \rho$ is a regular substitution$\}$, where f_1 and f_0 are 0-ary operations with $f_1 = \alpha$, $f_0 = \alpha^*$ for some $\alpha \in \Sigma_\omega$.

Let $\Omega_{\hat{\mathcal{P}}}$, $\Omega_{\hat{\mathcal{P}}}$, $\Omega_{\hat{\mathcal{E}}}$ and $\Omega_{\hat{\mathcal{F}}}$ be obtained by augmenting $\Omega_{\hat{\mathcal{RL}}}$ with $\{\cup\}$, $\{\cup, \cdot\}$, $\{*\}$ and $\{\cup, \cdot, *\}$ respectively. Here the binary operations $\cup$ and $\cdot$ are respectively the (set-theoretic) union and concatenation of languages $(L_1 \cdot L_2 = \{x_1 x_2 \mid x_1 \in L_1,\ x_2 \in L_2\})$. The Kleene star operation $*$ is a unary

operation defined by $L^* = \bigcup_{n=0}^{\infty} L^n$, where $L^0 = \{\lambda\}$ and $L^{i+1} = L \cdot L^i$.

The set $\Omega_{\mathfrak{S}}$ is extended further to $\Omega_{\hat{\mathfrak{R}}}$, [$\Omega_{\hat{\mathcal{A}}}$, $\Omega_{\hat{\mathfrak{X}}_1}$, $\Omega_{\hat{\mathfrak{X}}}$] by adding substitutions [nested iterated substitutions, single iterated substitutions, iterated substitutions].

For each set of operations $\Omega_{\hat{\mathfrak{X}}}$ just introduced (ALL,$\Omega_{\hat{\mathfrak{X}}}$) is an algebra. The corresponding algebraic closure operator on ALL will be denoted by $\hat{\mathfrak{X}}$. The several operators $\hat{\mathfrak{X}}$ have been defined in [16, 13, 2] in a different way (which clear up our notation) but both definitions are equivalent (cf. [2]). Each family closed under Π_0[Π] is called a [pre]quasoid [31, 1]. Let Q [PQ] denote the lattice of subalgebras of (ALL,Ω_{Π_0}) [(ALL,Ω_{Π})], i.e. the collection of all [pre]quasoids over Σ_ω. A family closed under $\hat{\mathfrak{U}}$[$\hat{\mathfrak{I}}$, $\hat{\mathfrak{P}}$, $\hat{\mathfrak{C}}$, $\hat{\mathfrak{F}}$, $\hat{\mathfrak{R}}$, $\hat{\mathcal{A}}$, $\hat{\mathfrak{X}}_1$, $\hat{\mathfrak{X}}$] is called a full trio [13] or M-AFL [full semi-AFL [22] or S-AFL, full pseudo-AFL [2] or P-AFL, full Kleene-AFL [2] or C-AFL, full AFL [16] or F-AFL, full substitution-closed AFL [17] or R-AFL, full super-AFL [23] or A-AFL, full hyper(1)-AFL [31, 10, 2] or H_1-AFL, full hyper-AFL [31, 8, 1] or H-AFL]. Let for X equal to M, S, P, C, F, R, A, H_1 or H, X-AFL denote the lattice of all full sub-X-AFL's of the full X-AFL ALL. So X-AFL is the collection of all full X-AFL's over Σ_ω.

We are now ready for the main result of this section.

Theorem 5.1. (1) PQ is a strongly algebraic lattice but not a GG- or a GH-lattice. Q, M-AFL and C-AFL are GH-lattices. In all these lattices we have $\bigwedge\{K_i \mid i \in I\} = \bigcap\{K_i \mid i \in I\}$, $\bigvee\{K_i \mid i \in I\} = \bigcup\{K_i \mid i \in I\}$, and T = ALL. The closure operators Π, Π_0, $\hat{\mathfrak{U}}$ and $\hat{\mathfrak{C}}$ are completely additive. (2) For X equal to S, P, F, R, A, H_1 or H, X-AFL is a GG-lattice with T = ALL.

Proof. First, note that each $\Omega_{\hat{\mathfrak{X}}}$ involved is countable and contains compact

and monotone operations only. Apart from $\hat{\mathfrak{X}} = \mathbb{H}$, in all cases covered by 5.1 $(ALL, \Omega_{\hat{\mathfrak{X}}})$ is a GG-algebra with languages as objects and $f_0 = \alpha^*$ as central object. Following 4.3 the corresponding lattice of subalgebras is a GG-lattice. Since $\Omega_{\mathbb{H}}$, $\Omega_{\mathbb{H}_0}$, $\Omega_{\hat{\mathfrak{AL}}}$ and $\Omega_{\hat{\mathfrak{C}}}$ only contain unary and 0-ary operations PQ, Q, M-AFL and C-AFL are strongly algebraic lattices (and the corresponding closure operators are completely additive; cf. 2.8). Consequently, the latter three are even GH-lattices.

PQ possesses an element, viz. REG, having no incomparable element and properly lying between $\perp$ and $\top$ [1]. By 3.3(2) PQ is not a GG- or a GH-lattice. □

By 5.1 results on incomparable elements in GG- and GH-lattices become applicable to quasoids and full X-AFL's (X equals M, S, P, C, F, R, A, H_1 or H). We leave the formulation of 3.3, 3.5 and 3.7 in terms of these structures to the reader.

With respect to the zero of the lattices mentioned in 5.1, we have that $\perp$ equals FIN in PQ [1], REG in Q [31, 1] and in X-AFL for $X \in \{M,C,P,S,F,R\}$ [16, 17, 2], the family of context-free languages in A-AFL [23], $\hat{\mathfrak{L}}_1(FIN)$ in H_1-AFL [2, 10], and the family of ETOL-languages in H-AFL [8, 2, 32].

We show that an improvement of 5.1(2) to GH-lattices in the cases X equals S, P or F is impossible. The problem whether X-AFL is (completely) distributive is open in the cases $X \in \{R, A, H_1, H\}$.

Theorem 5.2. Let X be equal to S, P or F.

(1) X-AFL is a non-distributive GG-lattice.

(2) X-AFL is not a GH-lattice.

(3) $\hat{\mathfrak{X}}$ is not a (completely) additive closure operator.

Proof. (1) Ginsburg and Spanier [18] established the existence of a full F-AFL, which is not coprime and does not split (Their argument also applies in the cases X equals S or P). According to 2.5 X-AFL is not distributive. Now (2) follows from (1), 2.8(2) and 4.2, whereas (3) is a consequence of (1) and 2.7(2). □

According to 3.5, 4.3 and 5.1 strongly compact full X-AFL's play an essential role in establishing the existence of largest incomparable full X-AFL's. So we now focus our attention to finitely generated and coprime full X-AFL's.

From 4.1 it follows that the weakly compact elements in X-AFL [Q, PQ] are precisely the countable full X-AFL's [quasoids, prequasoids]. For the compact elements, i.e. the finitely generated algebras we have a result similar to the cases $X \in \{S,F\}$ established in [15, 13] (A cyclic full F-AFL is usually called full principal).

Theorem 5.3. Let X be equal to S, P, F, R, A, H_1 or H. Then a full X-AFL is finitely generated if and only if it is cyclic. □

Remark that 5.3 does not hold for full C-AFL's, M-AFL's or (pre)quasoids, i.e. there exist finitely generated full C-AFL's, M-AFL's and (pre)quasoids that are not cyclic.

In the remaining part of this section we compare the power of different sets of operations. So in order to avoid any confusion we use prefixed phrases like X-cyclic, X-coprime, etc. (Note that F-coprime is called fully prime in [18]). We first consider the question whether an X-cyclic full X-AFL is also X'-cyclic. The proof of the following is straightforward.

Proposition 5.4. Let $\Omega_{\hat{\mathfrak{X}}} \subseteq \Omega_{\hat{\mathfrak{X}}'}$.

(1) If K is an X-cyclic full X-AFL, then $\hat{\mathfrak{X}}'(K)$ is X'-cyclic

(2) The following are equivalent.

(i) Each X'-cyclic full X'-AFL is X-cyclic.

(ii) If K is an X-cyclic full X-AFL, then $\hat{\mathfrak{X}}'(K)$ is X-cyclic. □

Note that we defined the sets of operations in a way such that $\Omega_{\hat{\mathfrak{X}}} \subseteq \Omega_{\hat{\mathfrak{X}}'}$ iff X-AFL $\supseteq$ X'-AFL. Thus the pairs (X, X') satisfying 5.4(1) can be determined from the inclusiondiagram in Figure 2 (solid lines) which has been established in [2].

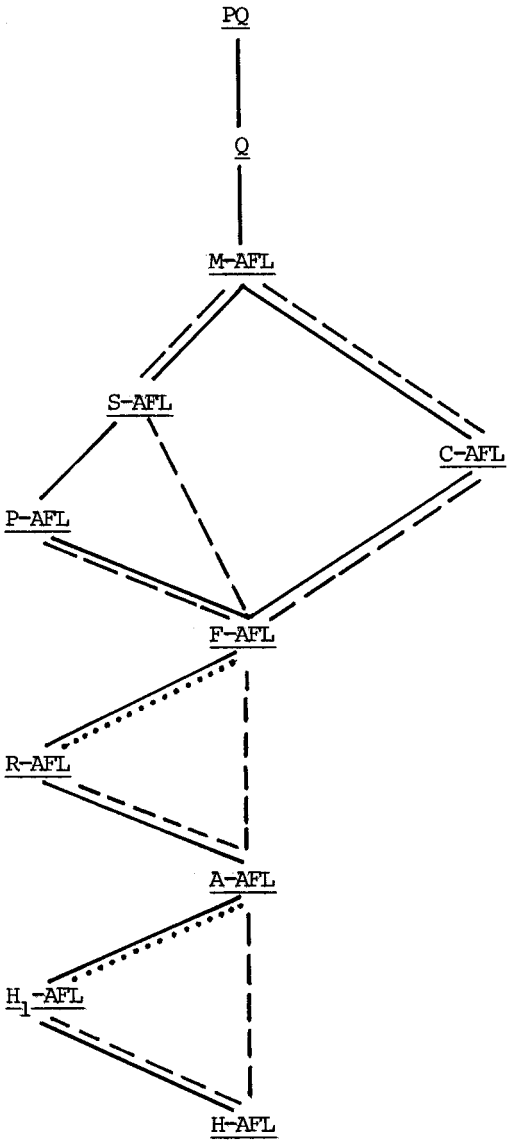
We write X $\sim$ X' when the pair (X, X') satisfies 5.4(2-i) (or 5.4 (2-ii)), and X $\not\sim$ X' for the negation of X $\sim$ X'. Remark that $\sim$ is a reflexive and transitive (but not a symmetric) relation satisfying: if X $\sim$ X' and X' $\not\sim$ X", then X $\not\sim$ X".

Theorem 5.5. (1) M $\sim$ S, (2) S $\sim$ F, (3) M $\sim$ C, (4) C $\sim$ F, (5) F $\sim$ A, (6) A $\sim$ H, (7) F $\not\sim$ R, (8) R $\sim$ A, (9) A $\not\sim$ H_1, (10) H_1 $\sim$ H, (11) P $\sim$ F.

Proof. The statements follow from 5.4 and respectively (1) [13] Proposition 5.1.1; (2-4) [15] Theorem 3.1 or [13] Theorem 5.4.1, and [2] Theorem 4.4; (5) [23] Theorem 4.7; (6) [11]; (7) [22] Theorem 3.2, [13] p. 239, [24]; (8) by (5); (9) [10]; (10) by (6); (11) by (2). □

These results are summarized in Figure 2 where X $\sim$ X' [X $\not\sim$ X'] is represented by a dashed [dotted] line connecting X and X'. The question whether S $\sim$ P is open.

Finally, we turn to coprime elements. From [18] we quote


PQ
Q
M-AFL
S-AFL
C-AFL
P-AFL
F-AFL
R-AFL
A-AFL
H_1-AFL
H-AFL

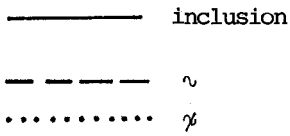

inclusion
∿
𝛾

Figure 2.

Theorem 5.6. (1) Each full F-AFL closed under $\hat{\mathfrak{R}}$ is F-coprime.

(2) Let K be an F-cyclic full F-AFL closed under $\hat{\mathfrak{R}}$ and unequal to REG.

(i) There exists a largest full F-AFL K' incomparable with K. Moreover, K' is closed under $\hat{\mathfrak{R}}$.

(ii) There exists a largest proper full sub-F-AFL K_ℓ of K. Furthermore K_ℓ is closed under $\hat{\mathfrak{R}}$ and K_ℓ equals the set of non-F-generators of K. □

Remark that 5.6(2-ii) is a special case of a general result on so-called "syntactic operators" due to Greibach ([24] Theorem 3.3). It is likely that 5.6(2-i) also allows a similar extension in terms of these operators.

By 5.6 each F-cyclic full H_1-AFL [A-AFL, H-AFL] K [unequal to the context-free languages, ETOL-languages] possesses a largest incomparable full F-AFL K' closed under $\hat{\mathfrak{R}}$; the question remains however, whether K' is also closed under $\hat{\mathfrak{M}}_1[\hat{\mathcal{A}}, \hat{\mathfrak{M}}]$.

6. Conclusion

In the previous sections we generalized theorems on incomparable full AFL's due to Ginsburg and Spanier [18] to lattice theory and universal algebra. Whereas some (straightforward) results can already be established for complete (2.1 - 2.3) or algebraic lattices (2.4) and universal algebras (2.9 - 2.10), the main results of this paper have only been proved for GG-lattices (Section 3) and GG-algebras (Section 4).

The most striking point of this extension from AFL-theory to abstract algebra is —— in our opinion —— that it makes clear what little AFL-

properties (abstracted in the notion of GG-algebra) one actually needs in order to establish Ginsburg and Spanier-like results. It also shows e.g. that in 2.2(2) (cf. 2.10) the "non-splitting" condition (introduced in [24]) is rather obvious. According to 2.2(1) there exists a maximal element c_b with $b \leq c_b < c$ and the additional requirement that c does not split only forbids c_b to possess an incomparable element in $\{x \mid \perp \leq x \leq c\}$.

This brings us to another interesting problem. Apart from the obvious question whether the definitions of GG-lattice and GG-algebra could be weakened while 3.3, 3.5, 3.7 and 4.3 still remain valid, we consider a further restriction. Let a relative GG-lattice be an algebraic lattice L satisfying (G1) and

(G2R) For each not weakly compact element t in L and for each pair of weakly compact elements a and b with $a,b < t$ there exists a weakly compact element c in K such that $a < c < t$ and $c \wedge b = a \wedge b$.

It follows almost directly that for each not weakly compact element t of a relative GG-lattice the set $\{x \mid \perp \leq x \leq t\}$ is also a relative GG-lattice. The question naturally arises which additional conditions one should require on a GG-algebra A in order that Sa(A) becomes a relative GG-lattice. Results in this direction would certainly have applications in AFL-theory (cf. [18] Section 4).

On the other hand particular AFL-structures can serve as (counter)example in lattice theory. As an example we show the correctness of the diagram in Figure 1. First, we note that the right-upper part (consisting of the square AL, CL, CDCL and CDAL) is well known [5, 6, 28]. By 2.6 we have SAL = CDAL, and 3.3(1) implies that the inclusions GGL $\subseteq$ AL and GHL $\subseteq$ SAL are proper. From 5.1(1) and 5.2 it follows that SAL $\not\subseteq$ GGL, GGL $\not\subseteq$ SAL, and that the inclusion GHL $\subseteq$ GGL is also proper.

References

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

2. P.R.J. Asveld, Extensions of language families and canonical forms for full AFL-structures, TW-memo No. 167, Twente University of Technology, Enschede, The Netherlands, 1977.

3. V.K. Balachandran, A characterization of $\Sigma\Delta$-rings of subsets, Fund. Math. 41 (1954) 38 - 41.

4. V.K. Balachandran, On complete lattices and a problem of Birkhoff and Frink, Proc. Amer. Math. Soc. 6 (1955) 548 - 553.

5. R. Balbes, Ph. Dwinger, "Distributive Lattices", University of Missouri Press, Columbia, Missouri, 1974.

6. G. Birkhoff, "Lattice Theory", 3rd ed., Amer. Math. Soc., Providence, R.I., 1967.

7. P.M. Cohn, "Universal Algebra", Harper & Row, New York, 1965.

8. P.A. Christensen, Hyper-AFL's and ETOL systems, pp. 254 - 257 in: G. Rozenberg, A. Salomaa (Eds.),"L Systems", Lecture Notes in Computer Science, Vol. 15, Springer, Berlin-Heidelberg-New York, 1974.

9. P. Crawley, R.P. Dilworth, "Algebraic Theory of Lattices", Prentice Hall, Englewood Cliffs, N.J. 1973.

10. J. Engelfriet, Iterating iterated substitution, Theor. Comp. Sci. 5 (1977) 85 - 100.

11. J. Engelfriet, G. Slutzki, Personel communication.

12. L. Geissinger, W. Graves, The category of complete algebraic lattices, J. Comb. Th. (A) 13 (1972) 332 - 338.

13. S. Ginsburg, "Algebraic and Automata-Theoretic Properties of Formal Languages", North-Holland, Amsterdam, 1975.

14. S. Ginsburg, J. Goldstine, Intersection-closed full AFL and the recursively enumerable languages, Inform. Contr. 22 (1973) 201 - 231.

15. S. Ginsburg, S.A. Greibach, Principal AFL, J. Comp. System Sci. 4 (1970) 308 - 338.

16. S. Ginsburg, S.A. Greibach, J.E. Hopcroft, Studies in Abstract Families of Languages, Mem. Amer. Math. Soc. 87 (1969).

17. S. Ginsburg, E.H. Spanier, Substitution in families of languages, Inform. Sci. 2 (1970) 83 - 110.

18. S. Ginsburg, E.H. Spanier, On incomparable Abstract Families of Languages (AFL), J. Comp. System Sci. 9 (1974) 88 - 108.

19. G. Grätzer, On the family of certain subalgebras of a universal algebra, Indag. Math. 27 (1965) 790 - 802 (also: Proc. Nederl. Akad. Wetensch. (A) 68 (1965) 790 - 802).

20. G. Grätzer, "Universal Algebra", Van Nostrand, Princeton, N.J., 1968.

21. G. Grätzer, "Lattice Theory: First Concepts and Distributive Lattices", Freeman, San Francisco, Ca., 1971.

22. S.A. Greibach, Chains of full AFL's, Math. Systems Theory 4 (1970) 231 - 242.

23. S.A. Greibach, Full AFL's and nested iterated substitution, Inform. Contr. 16 (1970) 7 - 35.

24. S.A. Greibach, Syntactic operators on full semi-AFL's, J. Comp. System Sci. 6 (1972) 30 - 76.

25. P.R. Halmos, "Naive Set Theory", Springer, Berlin-Heidelberg-New York, 1960.

26. B. Jónsson, "Topics in Universal Algebra", Lecture Notes in Mathematics 250, Springer, Berlin-Heidelberg-New York, 1972.

27. A.I. Mal'cev, "Algebraic Systems", Springer, Berlin-Heidelberg-New York, 1973.

28. G.N. Raney, Completely distributive complete lattices, Proc. Amer. Math. Soc. 3 (1952) 677 - 680.

29. A. Salomaa, "Formal Languages", Academic Press, New York, 1973.

30. E.T. Schmidt, "Kongruenzrelationen algebraischen Strukturen", VEB, Berlin, 1969.

31. J. van Leeuwen, F-iteration languages, Memorandum, University of California, Berkeley, Ca., 1973.

32. J. van Leeuwen, A study of complexity in hyper-algebraic families, pp. 323 - 333 in: A. Lindenmayer, G. Rozenberg (Eds.), "Automata, Languages, Development", North-Holland, Amsterdam, 1976.

Footnote

[1] From 4.2(iv) it follows that each GG-algebra has an uncountable carrier set. Remark that a subalgebra of a GG-algebra A is not a GG-algebra in general. Particularly, each (weakly) compact element in Sa(A) is not a GG-algebra. Nevertheless, each subalgebra of A satisfies $.2(i), (ii) and (iii).

S A M E N V A T T I N G

In dit proefschrift worden gegeneraliseerde (kontekst-onafhankelijke) grammatika's onderzocht, dat wil zeggen grammatika's met een aftelbare in plaats van een eindige verzameling produktieregels, die bovendien aan de volgende voorwaarde voldoen: alle rechterleden van regels met hetzelfde linkerlid vormen tezamen een taal uit een gegeven familie K van talen. Uiteraard hangt de familie van talen voortgebracht door deze gegeneraliseerde grammatika's (van een gegeven type) af van K. In het bizonder wordt aandacht besteed aan de operaties op talen, die door dergelijke gegeneraliseerde grammatika's geïnduceerd worden, en aan de bijbehorende transformaties op families talen en hun dekpunten.

Afgezien van een inleiding (Hoofdstuk 1) bestaat dit proefschrift uit drie delen.

Deel I (Hoofdstukken 2, 3 en 4) behandelt generalisaties van grammatika's. In Hoofdstuk 2 (verschenen in *Information and Control 34 (1977) 248 - 269*) beschouwen we generalisaties van (gestuurde) ETOL systemen. Het belangrijkste resultaat in dit hoofdstuk bestaat uit voldoende voorwaarden voor K en de besturing, opdat de familie van talen voortgebracht door deze gegeneraliseerde systemen een "volledige abstrakte familie talen" ("full AFL") vormt, die gesloten is onder geïtereerde substitutie. In Hoofdstuk 3 (verschenen in *Acta Informatica 8 (1977) 285 - 302*) worden generalisaties van deterministische ETOL systemen bestudeerd. Onder iets strengere voorwaarden op de besturing kunnen soortgelijke (maar wezenlijk verschillende) stellingen worden bewezen als in Hoofdstuk 2. Generalisaties van (gestuurde) lineaire macro grammatika's worden in Hoofdstuk 4 ingevoerd. Van deze grammatika's beschouwen we twee varianten en we bewijzen dat de ene overeenkomt met de grammatika's uit Hoofdstuk 2, terwijl de andere gelijkwaardig is met de grammatika's uit Hoofdstuk 3. Het tweede belangrijke resultaat uit Hoofdstuk 4 geeft het verband

tussen deze grammatika's en een klasse van niet-deterministische programma-schema's (zonder testen), register programma's genaamd.

In Deel II (Hoofdstukken 5 en 6) beschouwen we de transformaties op families talen, die geïnduceerd worden door operaties op talen behorende bij gegeneraliseerde grammatika's (nl. die uit Deel I en een paar bizondere gevallen zoals kontekst-vrije K-grammatika's en reguliere K-grammatika's). In Hoofdstuk 5 bestuderen we de dekpunten van deze en aanverwante transformaties en we karakteriseren hen in termen van typen "volledige abstrakte families talen". Hoofdstuk 6 behandelt enkele (eindige) partiëel geordende monoïden voortgebracht door deze transformaties. De struktuur van deze monoïden verschaft ons inzicht in de (on)afhankelijkheid van afsluitingseigenschappen.

In Deel III (Hoofdstuk 7) wordt met behulp van universele algebra en tralie-theorie de struktuur van de verzameling van alle dekpunten behorende bij deze transformaties nader onderzocht.

Errata and Remarks

1 Errata

ERRATA			
Page	**Line**	**Change**	**Into**
11	+13	$\mathcal{A}$	$\hat{\mathcal{A}}$
18	+14	K-substitution	REG-substitution
22	−5	$\mathrm{LB_{OI}}$ $\mathrm{LB_{IO}}$	$\mathrm{L_{OI}}$ $\mathrm{L_{IO}}$
69	+15	(V, T, I_0, U_0, S)	(V, T, I_0, U_0, M_0, S)
94	+1	that	this
96	+1	},	}, $\psi_i \in \Psi_n$ $(1 \leq i \leq k)$,
96	−7	a	an
97	+11	t'_n, $(n > 1)$	t'_n $(n > 1)$,
101	−6	;	,
150	−12	in K	in K_∞
171	+13	or	of
171	−9	operstors	operators
172	+14	repectively	respectively
174	+11	$\Gamma(A)$	$\Gamma(a)$
176	−7	σ-SIMPLE	$\underline{\sigma\text{-SIMPLE}}$
200	+13	exists	exist
222	+8	inclusiondiagram	inclusion diagram
225	+14	c in K	c in L
228	−1	\$.2(i)	4.2(i)

2 Remarks

Chapter 4 appeared as

• P.R.J. Asveld & J. Engelfriet, Extended Linear Macro Grammars, Iteration Grammars, and Register Programs, *Acta Inform.* **11** (1979) 259-285.

Chapter 7 appeared in slightly modified form as

• P.R.J. Asveld, An Algebraic Approach to Incomparable Families of Formal Languages, pp. 455–475 in G. Rozenberg & A. Salomaa (Eds.): *Lindenmayer Systems — Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology* (1992), Springer-Verlag, Berlin – Heidelberg – New York.

Related material can be found in

• P.R.J. Asveld, Space-Bounded Complexity Classes and Iterated Deterministic Substitution, *Inform. Contr.* **44** (1980) 282-299

• P.R.J. Asveld, On Controlled Iterated GSM Mappings and Related Operations, *Revue Roum. Math. Pures Appl.* **XXV** (1980) 139-145

• P.R.J. Asveld, Time and Space Complexity of Inside-Out Macro Languages, *Internat. J. Comput. Math.* **10** (1981) 3-14

• P.R.J. Asveld & J. Engelfriet, A Note on Non-Generators of Full AFL's, *Internat. J. Comput. Math.* **12** (1982) 13-17

• P.R.J. Asveld, Complete Symmetry in D2L Systems and Cellular Automata, *Internat. J. Comput. Math.* **19** (1986) 211-223

• P.R.J. Asveld, Complexity Aspects of Iterated Rewriting — A Survey, pp. 89–105 in P.R.J. Asveld & A. Nijholt (Eds.): *Essays on Concepts, Formalisms, and Tools — A Collection of Papers Dedicated to Leo A.M. Verbeek* (1987), C.W.I. Tract **42**, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

• P.R.J. Asveld, Abstract Grammars Based on Transductions, *Theoret. Comput. Sci.* **81** (1991) 269-288

• P.R.J. Asveld, An Infinite Sequence of Full AFL-structures, Each of Which Possesses an Infinite Hierarchy, pp. 175–186 (Chapter 15) in C. Martin-Vide & V. Mitrana (Eds.), *Where Mathematics, Computer Science, Linguistics and Biology Meet* (2001), Kluwer, Dordrecht, The Netherlands

Theorem 15.6 in this latter paper generalizes one of the main results of Chapter 2: viz. Theorem 4.4 (p. 50).