MACROS,

ITERATED SUBSTITUTION AND

LINDENMAYER AFL's

by

Arto Salomaa

DAIMI PB-18
Oktober 1973

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55

MACROS, ITERATED SUBSTITUTION AND LINDENMAYER AFL's

by

Arto Salomaa

Aarhus University
1973

1. Introduction and basic definitions

The purpose of this paper is to extend and make more explicit some notions and results presented in [5], [6], [9], and [10]. The reader is assumed to be familiar with the standard theory of L-systems, [4] or [8]. In particular, we use the customary abbreviations for various types of OL-systems: D(deterministic), P (propagating, with no erasing rules), E (extended system, intersection with the set of words over a terminal alphabet is allowed), and T (system with tables).

One of the first observations concerning L-systems was that the corresponding language families have very weak closure properties, in fact, many of the families are anti-AFL's, i.e., closed under none of the AFL operations. However, this phenomenon is due to the lack of a terminal alphabet rather than to parallellism which is the essential feature concerning L-systems. E.g., the family TOL of all TOL-languages is an anti-AFL, whereas the family ETOL is a full AFL. Later on we will see how L-systems can be used to convert language families with weak closure properties into full AFL's in a rather natural way.

The basic notion in this paper, K-iteration grammar, is a slight generalization of the notion introduced by van Leeuwen [5]. The motivation for such a notion is three-fold:

i) It provides a uniform framework for discussing OL-systems and all of their context-free generalizations.

ii) It shows the relation between OL-systems and (iterated) substitutions.

iii) It associates with each family K of languages (having certain mild closure properties) some full AFL's, obtained from K in the "Lindenmayer way".

We make the following conventions, valid throughout this paper. All language families discussed are non-trivial, i.e., they contain at least one language containing a nonempty word. (A language family is understood as in [8].) Two generative devices are termed <u>equivalent</u> if they generate the same language or else the language generated by one device differs from the language generated by the other through the empty word λ. (Thus in this sense, for any context-free grammar, there is an equivalent context-free grammar with no λ-rules.)

We introduce first some standard terminology and notations. Let K be a family of languages. A K-substitution is a mapping δ from some alphabet V into K. The mapping δ is extended to languages over V in the usual fashion. For language families K_1 and K_2, we define

(1) $\quad \mathrm{Sub}(K_1, K_2) = \{\delta(L) \mid L \in K_2 \text{ and } \delta \text{ is a } K_1\text{-substitution}\}$.

If K_2 = OL or K_2 = TOL, families (1) are called macro-OL and macro-TOL families, respectively, and denoted by K_1MOL and K_1MTOL. Macros were introduced in [1] and [2], where especially the cases K_1 = F (the family of finite languages) and K_1 = R (the family of regular languages) were investigated. Using the fact (cf. [7]) that the family of EOL-languages is closed under arbitrary homomorphism, it is easy to show that

$$\mathrm{FMOL} = \mathrm{EOL}.$$

(There seems to be no short direct proof for the inclusion FMOL $\subseteq$ EOL.) Similarly, one can prove that

$$\mathrm{FMTOL} = \mathrm{RMTOL} = \mathrm{ETOL}.$$

On the other hand, FMOL is properly included in RMOL because Herman's language

$$h^{-1}\{a^{2^n} \mid n \geq 0\} \quad \text{with } h(a) = a,\ h(b) = \lambda,$$

is in the difference RMOL-FMOL, cf. [3]. The family RMOL is the smallest full AFL (and the smallest AFL) including the family OL, cf. [2] or [7]. It is also the closure of FMOL under inverse homomorphism.

We will now present the basic definition. Let K be a family of languages. A K-iteration grammar is a quadruple $G = (V_N, V_T, S, U)$, where V_N and V_T are disjoint alphabets (of nonterminals and terminals), $S \in V^+$ with $V = V_N \cup V_T$ (initial word) and $U = \{\delta_1, \ldots, \delta_n\}$ is a finite set of K-substitutions defined on V and with the property that, for each i and each $a \in V$, $\delta_i(a)$ is a language over V. The language generated by such a grammar is defined by

(2) $L(G) = \cup \ \delta_{i_1} \ldots \delta_{i_k}(S) \cap V_T^*$,

where the union is taken over all integers $k \geqq 1$ and over all ordered k-tuples $(i_1, \ldots, i_k)$ with $1 \leqq i_j \leqq n$. The family of languages generated by K-iteration grammars is denoted by K_{iter}. By $K_{iter}^{(1)}$ we denote the subfamily of K_{iter}, consisting of languages generated by such grammars, where U consists of one element only.

The different OL-families can now be easily characterized within this framework. Consider the special case K = F. Then

$$K_{iter}^{(1)} = F_{iter}^{(1)} = EOL = FMOL.$$

(Note that it suffices to choose, for each $a \in V$, $\delta(a)$ to be the language consisting of the right sides of the productions with a on the left side.) Similarly,

$$F_{iter} = ETOL \ (=FMTOL = RMTOL).$$

The families with D and/or P are characterized as follows. D means that the δ's are homomorphisms, P means that the δ's are λ-free. Thus, EPDTOL is the subfamily of F_{iter}, obtained by such grammars where all substitutions δ are λ-free homomorphisms.

If one wants to consider families without E (OL, TOL, etc.), then one simply assumes that V_N is empty (which means that the intersection with V_T^* in (2) is superfluous). Note that in the general case the generative capacity is not affected by assuming that $S \in V_N$. Finally, the macro-families KMOL and KMTOL are obtained by K-iteration grammars satisfying the following condition. There is a sub-alphabet V_I of V_N such that, for each i and each $a \in V_I$, $\delta_i(a)$ is a finite language over V_N. Furthermore, for each i and each $a \in V_T$, $\delta_i(a)$ is empty and, for each i and each $a \in V_N - V_I$, $\delta_i(a)$ is a language (in K) over the alphabet V_T. (Here it is assumed that K contains all finite languages.)

Thus, all context-free L-systems find their counterpart in this formalism. Note, however, that so far (apart from regular macros) one has not considered in the theory of L-systems cases more general than K = F.

2. How to get rid of erasing

In this section, we establish the basic tool needed in later proofs for closure results. We say that a K-iteration grammar is λ-free iff each of the substitutions δ_i is λ-free.

Theorem 1

Assume that K is a language family closed under finite substitution and intersection with regular languages, L is a language generated by a λ-free K-iteration grammar, and h is an arbitrary homomorphism. Then also h(L) (or $h(L) - \{\lambda\}$) is generated by a λ-free K-iteration grammar.

Proof

Let L be generated by a λ-free K-iteration grammar $G = (V_N, V_T, S, U)$. We first list a few assumptions that can be made without loss of generality:

i) $S \in V_N$ (this was noted already before), and none of the languages $\delta_i(a)$ contains a word where S occurs,

ii) For each $\delta_i \in U$ and each $a \in V_T$, $\delta_i(a)$ is empty.

This is achieved by taking what might be called the "synchronized version" of the original grammar: Consider first the whole set $V = V_N \cup V_T$ to consist of nonterminals, add a new terminal alphabet $V_T' = \{a' \mid a \in V_T\}$ and define

(3) $\delta_i'(a) = \delta_i(a) \cup \{a'\}$, $\delta_i'(a') = \emptyset$, $a \in V_T$.

If we now make an alphabetic change back to our original notation, ii) follows. Note that the languages on the right sides of (3) belong to K by our assumptions.

iii) If h maps V_T into V_1^* then $V \cap V_1 = \emptyset$. This is achieved by making, if necessary, an alphabetic change in G.

Consider now the following assertion:

iv) Theorem 1 is true for λ-free homomorphisms h.

This follows immediately by ii) and iii), because it suffices to take $\delta_i(a) = \{h(a)\}$, for $a \in V_T$, and $\delta_i(a) = \emptyset$, for $a \in V_1$ (the target alphabet of h).

We now divide V_T into two parts: V_{die} consists of those letters a for which $h(a) = \lambda$; V_{live} consists of the remaining letters of V_T. Because the case $V_{live} = \emptyset$ is trivial, we may assume by iv) that both of these parts are nonempty. If L is entirely over the alphabet V_{die}, there is nothing to prove. Therefore, we may assume:

v) L contains a word involving a letter of V_{live}.

Two somewhat more difficult reductions remain in the proof of Theorem 1. For each nonterminal $A \in V_N$, we denote by L(G, A) the language generated by the K-iteration grammar $G = (V_N, V_T, A, U)$. We say that $A \in V_N$ is <u>living</u> iff each word in L(G, A) contains a letter of V_{live}, <u>dying</u> iff $L(G, A) \subseteq V_{die}^*$, and <u>undecided, ot</u>herwise. We may obviously assume that, for all A, L(G, A) is nonempty. Note that by v) S cannot be dying. We now claim:

vi) There is no loss of generality in assuming that G has no undecided nonterminals.

To establish vi), we first prove that there is a λ-free K-iteration grammar G' equivalent to G such that, with the possible exception of the initial letter, G' has no undecided nonterminals.

The terminal alphabet of G' is V_T (i.e., coincides with that of G). The nonterminal alphabet of G' consists of letters $\bar{A}$, where A is a nonterminal of G, of letters $\bar{\bar{A}}$, where A is an undecided nonterminal of G, and of a new initial letter S_1. To define the substitutions δ_i' of G', we consider two finite substitutions t_1 and t_2, defined on V. For $A \in V_N$, $t_1(A) = \{\bar{A}\}$; $t_2(A) = \{\bar{A}, \bar{\bar{A}}\}$ or $t_2(A) = \{\bar{A}\}$, depending on whether or not A is undecided.

For $a \in V_T$, $t_1(a) = t_2(a) = \{a\}$.

Each δ_i' maps S_1 to $\bar{S}$ or to $\{\bar{S}, \bar{\bar{S}}\}$, depending on whether S is living or undecided. Each δ_i' maps every letter of V_T into the empty language. For every living letter $A \in V_N$,

$$\delta_i'(\bar{A}) = t_2(\delta_i(A)).$$

For every dying letter $A \in V_N$,

$$\delta_i'(\bar{A}) = t_1(\delta_i(A)).$$

To define δ_i' in the remaining cases, we consider two regular languages R_1 and R_2. R_1 consists of all words over V_{die} and, in addition, of all words over the total alphabet of G' where at least one nonterminal occurs and, moreover, all occurring nonterminals are of the form $\bar{A}$, for some dying or undecided letter $A \in V_N$. R_2 consists of all words over V_T which are not words over V_{die} and, in addition, of all words over the total alphabet of G' which contain at least one letter $\bar{A}$, for some living $A \in V_N$, or at least one letter $\bar{\bar{A}}$, for some undecided $A \in V_N$. Let now $A \in V_N$ be undecided. We define

$$\delta_i'(\bar{A}) = t_1(\delta_i(A)) \cap R_1,$$
$$\delta_i'(\bar{\bar{A}}) = t_2(\delta_i(A)) \cap R_2.$$

Having completed the definition of G', we first note that G' contains no undecided letters, with the possible exception of S_1. All nonterminals $\bar{A}$, where $A \in V_N$ is undecided or dying, are dying. All nonterminals $\bar{A}$, where $A \in V_N$ is living, as well as all nonterminals $\bar{\bar{A}}$, where $A \in V_N$ is undecided, are living. It is also easy to see that G and G' are equivalent. In fact, derivations according to G' are derivations according to G, with some bars added to the nonterminals. (Intuitively, the decision of making a living or dying contribution is made as early as possible in G'.)

If S_1 is living, we have established vi). Otherwise, we replace G' by the grammar obtained from G' by omitting S_1 and $\bar{S}$. Now the initial letter is $\bar{\bar{S}}$. (Here assumption i) should be remembered.) Although the new grammar might not be equivalent to G any more, it suffices for our purposes, since we are looking for a grammar for h(L) and the words obtained from $\bar{S}$ contribute at most the empty word to h(L).

According to vi), we from now on assume that G contains no undecided nonterminals. We cannot simply discard the dying nonterminals from G because they may influence the mechanism of synchronization. However, all information needed to know the effect of the dying letters is contained in the knowledge of which among them are present. Our first construction is based on this observation.

Let us denote by M_{live} the alphabet consisting of the letters of V_{live} and of the living nonterminals of G. Similarly, denote by M_{die} the alphabet consisting of the letters of V_{die} and of the dying nonterminals of G. Let

M_1 be the alphabet consisting of pairs $<A, B>$, where $A \in M_{live} - \{S\}$ and B is a subset of M_{die}.

Consider the language R_3 consisting of all words of the form

$$x_1 y_1 x_2 \dots x_k y_k x_{k+1} \quad , \quad k \geqq 1 ,$$

where each x_i is a word over the alphabet M_{die} and, for $1 \leqq i \leqq k-1$, y_i is a letter $<A, B>$ of M_1 such that B is the set of letters occurring in x_i, and y_k is a letter $<A, B>$ of M_1 such that B is the set of letters occurring in $x_k x_{k+1}$. Obviously, R_3 is regular. Consider then some substitution δ_i of the grammar G and some subset B of M_{die}. The language $R_4(i, B)$ is obtained from R_3 by changing each $y_1 = <A_1, B_1>$ to some $<A_1, B_1 \cup \varphi(B)>$ where $\varphi(B)$ ranges over subsets of M_{die} satisfying the following condition. If $d_1, \dots, d_r$ are the elements of B, then there is a word X in $\delta_i(d_1 \dots d_r)$ such that $\varphi(B)$ is the set of letters occurring in X. Obviously, also $R_4(i, B)$ is regular.

We also need two finite substitutions t_3 and t_4, defined as follows. The substitution t_3 maps each element $A \in M_{live}$ to the set of pairs $<A, B>$, where this element occurs as the first component; t_3 maps the elements of M_{die} to themselves. The substitution t_4 maps elements of M_1 to themselves and elements of M_{die} to the empty word λ.

We are now ready to define a λ-free K-iteration grammar G'' as follows. The total alphabet of G'' is $M_1 \cup \{S\}$, and the terminal alphabet V_T'' consists of letters $<A, B>$, where $A \in V_{live}$ and B is a subset of V_{die}. The initial letter of G'' is S. The substitutions δ_i'' are defined as follows:

$$\delta_i''(S) = t_4(t_3(\delta_i(S)) \cap R_3),$$
$$\delta_i''(<A, B>) = t_4(t_3(\delta_i(A)) \cap R_4(i, B)).$$

Because S and each A are living, all substitutions δ_i'' are λ-free. Let now h_1 be the homormophism mapping each letter $<A, B>$ where $A \in V_{live}$ to $h(A)$. Clearly, h_1 is λ-free. It is also easy to see that $h_1(L(G'')) = h(L) - \{\lambda\}$. (Note that G'' is able to simulate only such derivations according to G, where at each step all occurrences of a dying nonterminal A are replaced by the same word in $\delta_i(A)$, this being a consequence of the definition of $\varphi(B)$. But this is all we need because such a derivation is possible, and the only task of the dying nonterminals is to block some words from the terminal language. Any other derivation would block at least the same words.) But h_1 is λ-free and Theorem 1 now follows by the assertion iv).

The proof above is a modification of the corresponding proof by van Leeuwen, [7], for EOL-languages. It depends on the family K whether or not the proof is constructive.

Theorem 2

Assume that K is a language family closed under finite substitution and intersection with regular languages. Then for each K-iteration grammar, there is an equivalent λ-free K-iteration grammar.

Proof

Given an arbitrary K-iteration grammar, we first replace λ in each of the languages $\delta_i(a)$ by some new letter c, and define $\delta_i(c)$ to be the empty language. We then apply Theorem 1 to the resulting grammar (which clearly is λ-free) and to the homomorphism erasing c and leaving the other letters unchanged.

3. Lindenmayer AFL's

We are now in the position to establish some basic results concerning the families K_{iter} and $K^{(1)}_{iter}$. Following van Leeuwen [5], we say that a family of languages is a quasoid iff it is closed under finite substitution and intersection with regular languages and, furthermore, contains a language a^*, where a is a letter.

Thus, a quasoid always contains all regular languages. A cone (also called a full trio) is always a quasoid but not vice versa. The essential difference is that a cone is closed under regular substitution. Theorems 1 and 2 are valid for quasoids K. We have stated them in a little more general form to see that they are valid for the family of finite languages F.

Theorem 3

If K is a quasoid then K_{iter} and $K^{(1)}_{iter}$ are full AFL's.

Proof

We note first that Theorems 1 and 2 are valid if attention is restricted to the family $K^{(1)}_{iter}$. The proof is also exactly the same because we did not make any use of the number n of the substitutions δ_i. Also in the following argument there is no difference between the two cases, and so we restrict ourselves to K_{iter}.

By known AFL-theory (cf. [8]), it suffices to prove that K_{iter} is closed under union, star, regular substitution and intersection with regular languages.

Assume, that, for i = 1, 2, L_i is generated by a K-iteration grammar G_i with the initial letter S_i. Without loss of generality, we assume that the nonterminal alphabets of G_1 and G_2 are disjoint. To generate the union $L_1 \cup L_2$, it suffices to introduce a new initial letter S which is mapped to the language $\{S_1, S_2\}$ by all substitutions. To generate $L_1{}^*$, you introduce two new nonterminals S (the new initial letter) and S_1' with substitutions

$$\delta_i(S) = \{\lambda, SS_1'\}, \quad \delta_i(S_1') = \{S_1', S_1\}.$$

For the remaining two operations, we assume by Theorem 2 that the grammar G_1 generating L_1 is λ-free. (The eventual loss of the empty word does not matter because we have closure under union.) Let ρ be a regular substitution. We first make ρ λ-free by replacing λ in the languages $\rho(a)$ with some new letter c. We also apply ii) from the proof of Theorem 1 to G_1. We now proceed exactly as in iv) of the proof mentioned: define $\delta_i(a) = \rho(a)$. (Note that K contains the languages $\rho(a)$). Finally, we apply Theorem 1 to erase c.

Let R be a regular language, accepted by a finite automaton M. The alphabet V' of a λ-free K-iteration grammar G' generating $L_1 \cap R$ consists of an initial letter S and of all triples (s_i, A, s_j), where s_i and s_j are states of M and A is a letter of G_1. The terminal alphabet of G' consists of such triples where A is a terminal letter of G_1. For any two states s_i and s_j of M, let $R_1(i,j)$ be the language over the alphabet V' consisting of all words x (of length $\geqq 1$) satisfying each of the following conditions:

i) In the first letter of x, the first state symbol is s_i.

ii) In the last letter of x, the second state symbol is s_j.

iii) The second state symbol of any letter of x (except for the last letter) equals the first state symbol of the next letter of x.

It is immediate that $R_1(i,j)$ is a regular (in fact, a 2-testable) language. Let t be the finite substitution mapping each letter A of G_1 to the set of triples having A as their middle symbol. We are now ready to define the substitutions δ_k' of G'. Each δ_k' maps S to the set consisting of triples (s_0, S_1, s_F), where s_0 is the initial state of M and s_F ranges over the final states of M. In general, for any letter A of G_1, and any states s_i and s_j,

$$\delta_k(s_i, A, s_j) = t(\delta_k(A)) \cap R_1(i,j).$$

Let now h_1 be the literal homomorphism mapping the triple (s_i, A, s_j), where A is a terminal letter of G_1, to the letter A. Then

$$L_1 \cap R = h_1(L(G')).$$

Hence, Theorem 3 follows.

The following theorem, [5], [9], [10], can be established similarly.

Theorem 4

If K is a quasoid then the families KMOL and KMTOL are full AFL's.

Since the full AFL's associated with a quasoid K in Theorems 3 and 4 are obtained by parallel rewriting, they are naturally called Lindenmayer AFL's. Apart from the obvious inclusions

$$KMOL \subseteq KMTOL \subseteq K_{iter}\ ,\quad KMOL \subseteq K_{iter}^{(1)} \subseteq K_{iter}\ ,$$

very little is known about these AFL's, e.g., about the strictness of the inclusions.

We have considered only the families $K_{iter}^{(1)}$ and K_{iter}, obtained by one substitution and an arbitrary number of substitutions, respectively. A natural generalization is the family $K_{iter}^{(n)}$, obtained by grammars with at most n substitutions. (Thus, K_{iter} is the union of all these families.) For a quasoid K, all of these families are full AFL's.

It is a result by van Leeuwen [5] that $R_{iter}^{(1)}$ equals the family of languages accepted by pre-set push-down automata. It seems likely that this family is properly included in the family R_{iter} but we have no proof. To prove this, it suffices to show that the language

$$\{a^{2^m 3^n} \mid m, n \geqq 1\}$$

is not in EOL (van Leeuwen, personal communication). The family $FMOL = F_{iter}^{(1)}$ is not an AFL. The proof above fails with respect to regular substitution. One obtains only closure under finite substitution.

4. Hyper-AFL's

We now consider language families closed under iterated substitution. By definition, a hyper-AFL is a quasoid K such that $K_{iter} = K$. Thus by Theorem 3 every hyper-AFL is a full AFL. Moreover, it can be shown to be a super-AFL.

Theorem 5

The indexed languages form a hyper-AFL.

Proof

The proof of van Leeuwen [6] is almost directly applicable. The only difference is that production ii) in the proof has to be replaced by the productions

$$\overline{X}_0 \rightarrow \overline{X}_0 L_i \quad , \ i = 1, \ldots, n$$

where the L_i's are flags corresponding to the different substitutions δ_i.

By Theorem 5, the family R_{iter} (and also the family CF_{iter}) is contained in the family of indexed languages. On the other hand, R_{iter} includes all of the context-free extensions of OL-languages considered in the literature.

We mention, finally, the following interesting open problem: Does the proper inclusion $K \subset K_{iter}$ imply the proper inclusion $K_{iter} \subset (K_{iter})_{iter}$? This would give rise to an infinite hierarchy.

References

[1] K. Culik II, On some families of languages related to developmental systems. International Journal of Computer Mathematics, to appear.

[2] K. Culik II and J. Opatrny, Macro OL-systems. University of Waterloo, Department of Applied Analysis and Computer Science Report CS-73-12 (1973).

[3] G. Herman, Closure properties of some families of languages associated with biological systems. Information and Control, to appear.

[4] G. Herman and G. Rozenberg, Developmental Systems and Languages. North-Holland Publishing Co., to appear.

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

[6] J. van Leeuwen, Hyper-AFL's and all pre-set PDA-languages are indexed. Memorandum, University of California, Berkeley (1973).

[7] J. van Leeuwen, Pre-set pushdown automata and OL-grammars. University of California, Berkeley, Computer Science Technical Report 10 (1973).

[8] A. Salomaa, Formal Languages. Academic Press (1973).

[9] A. Salomaa, Lindenmayer AFL's. Manuscript (1973).

[10] A. Salomaa, On some recent problems concerning developmental languages. Proc. 1. GI Fachtagung, Automaten und Formale Sprachen, Bonn 1973, to appear.