Controlled Fuzzy Parallel Rewriting

Peter R. J. ASVELD

Department of Computer Science
Twente University of Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands
E-mail: `infprja@cs.utwente.nl`

Abstract. We study a Lindenmayer-like parallel rewriting system to model the growth of filaments (arrays of cells) in which developmental errors may occur. In essence this model is the fuzzy analogue of the derivation-controlled iteration grammar. Under minor assumptions on the family of control languages and on the family of fuzzy languages in the underlying iteration grammar, we show that (i) regular control does not provide additional generating power to the model, (ii) the number of fuzzy substitutions in the underlying iteration grammar can be reduced to two, and (iii) the resulting family of fuzzy languages possesses strong closure properties, viz. it is a full hyper-AFFL, i.e., a hyper-algebraically closed full Abstract Family of Fuzzy Languages.

1. Introduction

The original motivation to introduce Lindenmayer systems, or L-systems for short, consisted of modeling the development of filamentous organisms [15], [16]. The state space of each individual cell of such an organism is a finite set, symbolically represented as an alphabet V, and rewrite rules over V provide for the development of single cells. More precisely, a rule $\alpha \to w$ with $\alpha \in V$ and $w \in V^\star$, allows for a state change ($w \in V$, $w \neq \alpha$), a cell death ($w = \lambda$, λ is the empty word), or the splitting of a cell in more than a single off-spring ($| w | > 1$, where $| w |$ is the length of the string w). Starting from an initial filament, i.e. a string over V, and applying the rules for individual cells in parallel yields the global state of the filament after a discrete time step. Iterating this rewriting process shows the development of this filament as function of the discrete time parameter. From a mathematical point of view the set of rules is just a finite substitution over V that is applied iteratively to the initial string.

Subsequent contributions to the extension of this model resulted in the distinction between nonterminal and terminal symbols as in Chomsky phrase-structure grammars, in several sets of rules (several finite substitutions, also called *tables*) instead of just a single one, and numerous ways of restricting or regulating the parallel rewriting process. We refer the reader to [13], [21] for surveys of the early days of L-system theory; [13] is more elementary and devoted to biological applications, whereas [21] concentrates on mathematical properties. More recent developments and related approaches can be found in [7], [22], of which [7] treats derivation-controlled rewriting in general, whereas [22] shows a rich variety of results closely related to or inspired by L-systems.

The extension of the basic model with different sets of rules (a finite number of finite substitutions instead of a single one) stems from the observation that a filamentous organism might develop in a different way under different external conditions

[20]. A typical example is the difference between day and night; in that case we have two sets of rules, or tables, viz. a day table τ_d and a night table τ_n, each table being a finite substitution over the alphabet V. Closely related to this extension are the so-called derivation-controlled tabled L-systems in which the order of application is prescribed by a control language over the table names [10], [18], [1]. E.g. in order to obtain the right sequence of day, followed by night, followed by day, etc., a regular control language of the form $(\tau_d\tau_n)^\star\tau_d$ can be used, provided each sequence should start and end with the day table τ_d. Similarly, but on a larger time scale, the order of the four seasons can be described by a regular control language of the form $(\tau_{\text{spring}}\tau_{\text{summer}}\tau_{\text{autumn}}\tau_{\text{winter}})^\star\tau_{\text{spring}}$.

In this paper we introduce a further extension of this model which enables us to describe developmental errors. Such an error occurs when, instead of applying the correct rule $\alpha \rightarrow w$ from the table τ, the symbol α is replaced by a string w' with $w' \neq w$ and $\alpha \rightarrow w'$ is *not* a rule in τ. In such a situation the "quality" of this incorrect off-spring w' should be strictly less than the corresponding correct one and, consequently, the "quality" of the entire filament should also decrease by this developmental error. In addition we want that making two developmental errors is worse than a single error and, in general, that each additional developmental error should strictly decrease the "quality" of the filament under consideration.

But how do we measure the "quality" of a string or filament x derived by a controlled tabled L-system G? In traditional formal language theory there only are two possibilities, viz. (i) x belongs to the language $L(G)$ generated by G: its "quality" equals 100%, or (ii) x does not belong to $L(G)$: the "quality" of x is 0%. Clearly, there is no room for expressing statements like "x is slightly imperfect due to a minor developmental error" or "x has been severely damaged by a long sequence of considerable errors during its development". This lack of expressibility is, of course, due to restrictions in set theory: the membership function or characteristic function $\mu_{L(G)}$ of a set, or a language $L(G)$ in our case, has two possible values only: $\mu_{L(G)}(x) = 1$ if $x \in L(G)$, and $\mu_{L(G)}(x) = 0$ if $x \notin L(G)$. Thus, if $L(G) \subseteq \Sigma^\star$, then $\mu_{L(G)}$ is a mapping of type $\mu_{L(G)} : \Sigma^\star \rightarrow \{0, 1\}$.

Fortunately, using fuzzy sets and fuzzy languages we are able to express "qualities" different from 0% and 100%, since $\mu_{L(G)}$ is now a mapping of type $\mu_{L(G)} : \Sigma^\star \rightarrow \mathcal{L}$ where $\mathcal{L}$ is a complete lattice, eventually provided with additional operations and properties. As a typical example, the reader may consider the case in which $\mathcal{L}$ equals the real interval [0,1] with min and max as lattice operations. Fuzzy languages have been introduced in [17], which is restricted to fuzzy analogues of Chomsky grammars and languages. In [19] fuzzy Lindenmayer systems and their languages have been studied, however, without any motivation in terms of developmental errors. This motivation is the obvious parallel Lindenmayer variant based on the idea of grammatical error studied in [3], [4], [5].

So in fuzzy L-system theory the "quality" of a string is a value in $\mathcal{L}$ which might be anything in between 0 (the smallest element of $\mathcal{L}$) and 1 (the greatest element of $\mathcal{L}$) depending on the actual structure of $\mathcal{L}$. And making a developmental error in the derivation of x means that the "quality" of x will not increase compared to the previous string. But whether it will strictly decrease depends on the structure and the operations of $\mathcal{L}$ as well as their relation with the definition of derivation step; cf. Section 4 for details.

In dealing with developmental errors there is another problem. Usually, an L-system has in each of its tables a finite number of rewrite rules. Making a developmental mistake, i.e., replacing α by w' instead of by the correct string w can be modeled by adding the rule $\alpha \to w'$ to the table τ to which $\alpha \to w$ belongs, and requiring $\mu_{\tau(\alpha)}(w') < 1$, where $\tau(\alpha)$ is the set of all strings ω such that $\alpha \to \omega$ belongs to τ. This construction works for a finite number of possible developmental errors only. But, in general, there is an infinite number of ways to make mistakes, and filamentous development does not form an exception to this observation. So we should add an infinite number of rules $\alpha \to w'$ to τ or, equivalently, an infinite number of strings to the fuzzy set $\tau(\alpha)$. So each set $\{\omega \in \tau(\alpha) \mid 0 < \mu_{\tau(\alpha)}(\omega) < 1\}$ is allowed to be infinite. But then the language $\{\omega \in \tau(\alpha) \mid \mu_{\tau(\alpha)}(\omega) = 1\}$ might be infinite as well, or, equivalently, each $\tau(\alpha)$ may be a fuzzy subset of V^*, i.e., a fuzzy languages over V. However, we could not let be the sets $\tau(\alpha)$ arbitrary fuzzy languages over V: they should be restricted in some uniform way, otherwise we end up with languages $L(G)$ that are not even recursively enumerable; cf. [8]. A well-known way to restrict these fuzzy languages is the following: we require that each fuzzy language $\tau(\alpha)$ belongs to a given family K of fuzzy languages. The family K is a parameter in our approach: usually, we demand that K meets some minor conditions, but sometimes we simply take a concrete value for K, e.g., we take K equal to the family FIN_f of finite fuzzy languages.

This results in the notion of fuzzy K-iteration grammar which plays the main part in the present paper. Formally, such a grammar $G = (V, \Sigma, U, S)$ consists of an alphabet V, a terminal alphabet Σ ($\Sigma \subseteq V$), an initial symbol S ($S \in V - \Sigma$), and a finite set U of fuzzy K-substitutions over V. Thus for each τ in U, and for each α in V, $\tau(\alpha)$ is a fuzzy language over V that belongs to the family K. The controlled variant of this grammar concept is the so-called Γ-controlled fuzzy K-iteration grammar, or fuzzy (Γ, K)-iteration grammar where Γ is a family of (non-fuzzy) languages. A grammar $(G; M) = (V, \Sigma, U, S, M)$ of this type consists of a fuzzy K-iteration grammar (V, Σ, U, S) and a language M over U (considered as an alphabet) with $M \in \Gamma$. Each derivation D according to $(G; M)$ satisfies the condition that the sequence of fuzzy K-substitutions used in D constitutes a string in the control language M.

The remaining part of this paper is organized as follows. In Section 2 we introduce the basic notions with respect to fuzzy languages and operations on fuzzy languages. Section 3 is devoted to families of fuzzy languages. The formal definitions of fuzzy K-iteration grammar and of Γ-controlled fuzzy K-iteration grammar are provided in Section 4, where we also give a few examples of these grammars together with the fuzzy languages that they generate. Section 5 consists of some elementary but useful properties of fuzzy K-iteration and fuzzy (Γ, K)-iteration grammars. The main results, viz. Theorem 6.1 and its corollaries, which deal with the generating power of fuzzy (Γ, K)-iteration grammars, are in Section 6. Closure properties of the corresponding families of fuzzy languages are the subject of Section 7. Under minor conditions on the families Γ and K, the families $H_f(K)$ and $H_f(\Gamma, K)$ of fuzzy languages, generated by fuzzy K-iteration grammars and (Γ, K)-iteration grammars, respectively, possess strong closure properties very similar to the ones of the corresponding non-fuzzy language families; cf. [1]. Finally, Section 8 contains some concluding remarks.

2. Fuzzy Languages and Operations on Fuzzy Languages

We assume that the reader is familiar with basic formal language theory to the extend of the first few chapters of standard texts like [12], [14], [23]. L-systems and Abstract Families of Languages are treated much more thoroughly in [13], [21] and [9], respectively. Finally, we need some rudiments of lattice theory which can be found in most books on algebra; all what we use of lattice theory is also summarized in [2].

In order to define several types of fuzziness we need a few lattice-ordered structures. Instead of stacking adjectives, we collect some collections of properties under simple names as "type-*bb* lattice" for some short bit strings *bb*. The following definitions and examples are quoted from [5]. The definition of the principal notion of type 00-lattice is a slight modification of a structure originally introduced in [11].

Definition 2.1. An algebraic structure $\mathcal{L}$ or $(\mathcal{L}, \wedge, \vee, 0, 1, \star)$ is a *type-00 lattice* if it satisfies the following conditions.

- $(\mathcal{L}, \wedge, \vee, 0, 1)$ is a completely distributive complete lattice. Therefore for all a_i, a, b_i and b in $\mathcal{L}$, $a \wedge \bigvee_i b_i = \bigvee_i (a \wedge b_i)$ and $(\bigvee a_i) \wedge b = \bigvee_i (a_i \wedge b)$ hold. And 0 and 1 are the smallest and the greatest element of $\mathcal{L}$, respectively; so $0 = \bigwedge \mathcal{L}$ and $1 = \bigvee \mathcal{L}$.

- $(\mathcal{L}, \star)$ is a commutative semigroup.

- The following identities hold for all a_i's, b_i's, a and b in $\mathcal{L}$:

$$a \star \bigvee_i b_i = \bigvee_i (a \star b_i) \ ,$$

$$(\bigvee_i a_i) \star b = \bigvee_i (a_i \star b) \ ,$$

$$0 \wedge a = 0 \star a = a \star 0 = 0 \ ,$$

$$1 \wedge a = 1 \star a = a \star 1 = a \ .$$

A *type-01 lattice* is a type-00 lattice in which the operation $\star$ coincides with the operation $\wedge$; so it is a completely distributive complete lattice actually. A *type-10 lattice* is a type-00 lattice in which $(\mathcal{L}, \wedge, \vee, 0, 1)$ is a totally ordered set or chain, i.e., for all a and b in $\mathcal{L}$, we have $a \wedge b = a$ or $a \wedge b = b$. In a type-10 lattice the operations $\vee$ and $\wedge$ are usually denoted by max and min, respectively. Finally, when $\mathcal{L}$ is both a type-01 lattice and a type-10 lattice, $\mathcal{L}$ is called a *type-11 lattice.*

Example 2.2. As usual we denote the closed interval of all real numbers in between 0 and 1 by $[0, 1]$.
(1) The structure $([0, 1] \times [0, 1], \wedge, \vee, (0, 0), (1, 1), \star)$ in which the operations are defined by $(x_1, y_1) \wedge (x_2, y_2) = (\min\{x_1, x_2\}, \min\{y_1, y_2\})$, $(x_1, y_1) \vee (x_2, y_2) = (\max\{x_1, x_2\}, \max\{y_1, y_2\})$ and $(x_1, y_1) \star (x_2, y_2) = (x_1x_2, y_1y_2)$ for all x_1, x_2, y_1 and y_2 in $[0, 1]$ is a type-00 lattice.
(2) Consequently, $([0, 1] \times [0, 1], \wedge, \vee, (0, 0), (1, 1), \star)$ where the operations $\wedge$ and $\vee$ are defined as in (1) and $(x_1, y_1) \star (x_2, y_2) = (\min\{x_1, x_2\}, \min\{y_1, y_2\})$ for all x_1, x_2, y_1 and y_2 in $[0, 1]$, is a type-01 lattice.
(3) The structure $([0, 1], \min, \max, 0, 1, \star)$ with $x_1 \star x_2 = x_1x_2$ for all x_1 and x_2 in $[0, 1]$ is a type-10 lattice.
(4) Taking $\star$ equal to min in (3) yields a type-11 lattice.

The following useful fact is very easy to prove.

Lemma 2.3. *For each type-00 lattice $\mathcal{L}$, $a \star b \leq a \wedge b$ holds for all elements a and b in $\mathcal{L}$.*

Proof.
By the distributivity of $\star$ over $\vee$, $a \star (1 \vee b) = a \star 1 \vee a \star b$ holds. As $1 \vee b = 1$ and $a \star 1 = a$, we have $a = a \vee a \star b$, and therefore $a \star b \leq a$. Analogously, we obtain $a \star b \leq b$, and hence $a \star b \leq a \wedge b$. □

Of course, Lemma 2.3 implies that in a type-00 lattice the inequalities $a \star b \leq a$ and $a \star b \leq b$ also hold for all a and b.

Now we are ready to define fuzzy languages relative to the lattice-ordered structures of Definition 2.1.

Definition 2.4. Let $\mathcal{L}$ be a type-00 lattice and let Σ be an alphabet. A *$\mathcal{L}$-fuzzy language* over Σ is a $\mathcal{L}$-fuzzy subset of $\Sigma^\star$, i.e., it is a triple (Σ, μ_{L_0}, L_0) where μ_{L_0} is a function $\mu_{L_0} : \Sigma^\star \to \mathcal{L}$, the *degree of membership function*, and L_0 is the support of μ_{L_0}; i.e., $L_0 = \{w \in \Sigma^\star \mid \mu_{L_0}(w) > 0\}$. Very often we will write L_0 rather than (Σ, μ_{L_0}, L_0).

Henceforth, when $\mathcal{L}$ is clear from the context, we use "fuzzy language" instead of "$\mathcal{L}$-fuzzy language". Usually we write $\mu(x; L_0)$ instead of $\mu_{L_0}(x)$ in order to reduce the number of subscript levels.

For each fuzzy language L_0 over Σ, the *crisp language* $c(L_0)$ induced by L_0 —also known as the *crisp part* of L_0— is the subset of $\Sigma^\star$ defined by $c(L_0) = \{w \in \Sigma^\star \mid \mu(w; L_0) = 1\}$. Each ordinary (non-fuzzy) language L_0 coincides with its crisp part $c(L_0)$. Therefore an ordinary language will also be called a *crisp* language.

In dealing with fuzzy languages (Σ, μ_{L_0}, L_0) the degree of membership function μ_{L_0} is actually the principal concept, whereas the languages L_0, $c(L_0)$ and many other crisp languages like

$$L_{\geq a} = \{w \in \Sigma^\star \mid \mu(w; L_0) \geq a\} \ ,$$

$$L_{> a} = \{w \in \Sigma^\star \mid \mu(w; L_0) > a\} \ ,$$

$$L_{\leq a} = \{w \in \Sigma^\star \mid \mu(w; L_0) \leq a\} \ ,$$

$$L_{< a} = \{w \in \Sigma^\star \mid \mu(w; L_0) < a\} \ ,$$

$$L_{a \leq ; \leq b} = \{w \in \Sigma^\star \mid a \leq \mu(w; L_0) \leq b\} \ ,$$

where a and b are elements in $\mathcal{L}$, are derived notions.

Example 2.5. (1) Let $\mathcal{L}$ be the type-00 lattice of Example 2.2.(1). Consider the $\mathcal{L}$-fuzzy language L_0 over $\Sigma = \{a, b\}$ defined by

$$\mu(a^m b^n; L_0) = \left(\frac{m}{\max\{1,m,n\}}, \frac{n}{\max\{1,m,n\}}\right) \text{ if } m, n \geq 0.$$

In defining the degree of membership function is such a concrete case, we always tacitly assume that $\mu(x; L_0) = (0, 0)$ in all other, unmentioned cases for x in $\Sigma^\star$. Consequently, we have, e.g., $\mu(b^3a^2; L_0) = \mu(a^2ba^5; L_0) = \mu(ab^3a^2b^4; L_0) = (0, 0)$, etc.

Then the crisp part of L_0 equals $c(L_0) = \{a^m b^m \mid m \geq 1\}$; for each x in $c(L_0)$, we have $\mu(x; L_0) = (1, 1)$. Note that for each $m \geq 1$, $\mu(a^m; L_0) = (1, 0)$ and $\mu(b^m; L_0) = (0, 1)$, whereas for the empty word λ, we have $\mu(\lambda; L_0) = (0, 0)$.

(2) Now we take for $\mathcal{L}$ the type-10 lattice of Example 2.2.(3). Let L be the fuzzy language over $\{a, b\}$ defined by

$$\begin{array}{lll} \mu(w;L) = 0 & \text{if } \mid w \mid \ \neq \ 2^k \text{ for some } k \geq 0, \text{ and} \\ \mu(w;L) = 2^{-\#_b(w)} & \text{if } \mid w \mid \ = \ 2^k \text{ for some } k \geq 0. \end{array}$$

As usual, $\#_\sigma(w)$ denotes the number of times that the symbol σ occurs in the word w. Then $c(L) = \{a^{2^k} \mid k \geq 0\}$.

Throughout this paper we will restrict ourselves to the computable or even to the rational elements in $[0, 1]$. For an account on the impact of computability constraints in fuzzy formal languages we refer the reader to [8].

Starting from simple fuzzy languages we can define more complicated ones by means of operations on fuzzy languages. First, we consider the operations union, intersection and concatenation for fuzzy languages; they have been defined originally in [17] for the type-11 lattice $[0,1]$; cf. Example 2.2(4). In [4] we remarked that a generalization to the type-10 lattice of Example 2.2(3) is possible. However, it is straightforward to define these operations for arbitrary type-00 lattices; cf. [5] from which we cite the following definitions.

Let $(\Sigma_1, \mu_{L_1}, L_1)$ and $(\Sigma_2, \mu_{L_2}, L_2)$ be fuzzy languages, then the *union* of the fuzzy languages L_1 and L_2, denoted by $(\Sigma_1 \cup \Sigma_2, \mu_{L_1 \cup L_2}, L_1 \cup L_2)$ or abbreviated by $L_1 \cup L_2$, is defined by

$$\mu(x; L_1 \cup L_2) = \mu(x; L_1) \vee \mu(x; L_2) \ ,$$

for all x in $(\Sigma_1 \cup \Sigma_2)^\star$. And for the *intersection* of fuzzy languages L_1 and L_2, denoted by $(\Sigma_1 \cap \Sigma_2, \mu_{L_1 \cap L_2}, L_1 \cap L_2)$ or $L_1 \cap L_2$ for short, the equality

$$\mu(x; L_1 \cap L_2) = \mu(x; L_1) \wedge \mu(x; L_2) \ ,$$

holds for all x in $(\Sigma_1 \cap \Sigma_2)^\star$. Finally, for the *concatenation* of fuzzy languages L_1 and L_2, denoted by $(\Sigma_1 \cup \Sigma_2, \mu_{L_1 L_2}, L_1 L_2)$ or abbreviated to $L_1 L_2$, we have

$$\mu(x; L_1 L_2) = \bigvee \{\mu(y; L_1) \star \mu(z; L_2) \mid x = yz\}$$

for all x in $(\Sigma_1 \cup \Sigma_2)^\star$.

Example 2.6. Let $\mathcal{P}(X)$ denote the power set of the set X. Then $\mathcal{P}(\Sigma^\star)$ is the collection of all crisp languages over the alphabet Σ. Let $\mathcal{P}_f(\Sigma^\star)$ be the class of all fuzzy languages over Σ. Clearly, we have $\mathcal{P}(\Sigma^\star) = \{c(L) \mid L \in \mathcal{P}_f(\Sigma^\star)\}$. And $(\mathcal{P}_f(\Sigma^\star), \cap, \cup, \oslash, \Sigma^\star, \cdot)$ —where $\cap$, $\cup$ and $\cdot$ denote the operations union, intersection and concatenation for fuzzy languages, respectively— is *not* an example of a type-00 lattice, since $(\mathcal{P}_f, \cdot)$ is not a commutative semigroup. In case Σ contains a single letter only, $(\mathcal{P}_f, \cdot)$ is a commutative semigroup and $(\mathcal{P}_f(\Sigma^\star), \cap, \cup, \oslash, \Sigma^\star, \cdot)$ is a type-00 lattice. The same remarks apply to the structure $(\mathcal{P}(\Sigma^\star), \cap, \cup, \oslash, \Sigma^\star, \cdot)$ of crisp languages.

Once we have defined the operations of union and concatenation it is straightforward to define the operations of *Kleene* $+$ and *Kleene* $\star$ for a fuzzy language L; viz. by

$$L^+ = L \cup LL \cup LLL \cup \ldots = \bigcup \{L^i \mid i \geq 1\} \ , \qquad \text{and}$$

$$L^\star = \{\lambda\} \cup L \cup LL \cup LLL \cup \ldots = \bigcup \{L^i \mid i \geq 0\} \ ,$$

respectively, where $L^0 = \{\lambda\}$, and $L^{n+1} = L^n L$ with $n \geq 0$. In defining $L^\star$ we demand that $\mu(\lambda; L^\star) = 1$. Consequently, $L^\star = L^+ \cup \{\lambda\}$ where the latter set in this union is a crisp set.

Apart from these simple operations we need some other well-known ones, like homomorphisms and substitutions. They can be extended to fuzzy languages as well by means of the concept of fuzzy function; cf. [5] for the original definitions.

A *fuzzy relation* R between crisp sets X and Y is a fuzzy subset of $X \times Y$. If $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ are fuzzy relations, then their composition $R{\circ}S$ is defined by

$$\mu((x,z); R{\circ}S) = \bigvee\{\mu((x,y); R) \star \mu((y,z); S) \mid y \in Y\}. \tag{1}$$

A *fuzzy function* $f : X \to Y$ in its turn, is a fuzzy relation $f \subseteq X \times Y$, satisfying the condition that for all x in X: if $\mu((x,y); f) > 0$ and $\mu((x,z); f) > 0$ hold, then $y = z$ and hence $\mu((x,y); f) = \mu((x,z); f)$. For fuzzy functions (1) holds as well, but we write the composition of two functions $f : X \to Y$ and $g : Y \to Z$ as $g{\circ}f : X \to Z$ rather than as $f{\circ}g$.

As mentioned before, $\mathcal{P}(X)$ denotes the power set of the set X. In the sequel we need functions $f : V^\star \to \mathcal{P}(V^\star)$ that will be extended to $f : \mathcal{P}(V^\star) \to \mathcal{P}(V^\star)$ by $f(L) = \bigcup\{f(x) \mid x \in L\}$ and for each subset L of $V^\star$,

$$\mu(y; f(L)) = \bigvee\{\mu(x; L) \star \mu((x,y); f) \mid x \in V^\star\}. \tag{2}$$

Consequently, by (1) and (2) iterating a single fuzzy function f, yielding functions like $f{\circ}f$, $f{\circ}f{\circ}f$, and so on, are now defined. Clearly, each of these functions $f^{(n)}$ is of type $f^{(n)} : \mathcal{P}(V^\star) \to \mathcal{P}(V^\star)$. Of course, we can iterated a finite set of such functions $\{f_1, \ldots, f_n\}$ in the very same way.

3. Families of Fuzzy Languages

This section is devoted to some families of simple fuzzy languages, their crisp counterparts, and a few operators that transform families of fuzzy languages into other families. The next few definitions are simple generalizations based on well-known concepts for families of crisp languages; cf. [5].

Throughout this paper Σ_ω denotes a countably infinite set of symbols. All families of languages that we will consider in the sequel only use symbols from this set. Henceforth, $\mathcal{L}$ is a type-00 lattice, and "fuzzy" means "$\mathcal{L}$-fuzzy" actually.

Definition 3.1. A *family of fuzzy languages* K is a set of fuzzy languages (Σ_L, μ_L, L) such that each Σ_L is a finite subset of Σ_ω. As usual, we assume that for each fuzzy language (Σ_L, μ_L, L) in the family K, the alphabet Σ_L is minimal with respect to μ_L, i.e., a symbol α belongs to Σ_L if and only if there exists a word w in which α occurs and for which $\mu_L(w) > 0$ or, equivalently, for which $w \in L$ holds.

A family K of fuzzy languages is called *nontrivial* if K contains a language (Σ_L, μ_L, L) with $L \cap \Sigma_L^+ \neq \emptyset$, i.e., (Σ_L, μ_L, L) satisfies $\mu(x; L) > 0$ for some $x \in \Sigma_L^+$.

For each family K of fuzzy languages, the *crisp part* of K, denoted by $c(K)$, is defined by $c(K) = \{c(L) \mid L \in K\}$.

We already remarked that we write L rather than (Σ_L, μ_L, L) for members of a family of fuzzy languages. And we also assume that each family of fuzzy languages, that we will use in this paper, is closed under isomorphism ("renaming of symbols"), i.e., for each family K we assume that for each fuzzy language L in K over some alphabet Σ_L and for each bijective non-fuzzy mapping $i : \Sigma_L \to \Sigma_L'$ —extended to words and to languages in the usual way— we have that the language $i(L)$ also belongs to K. Consequently, we have the equality $\mu(x; L) = \mu(i(x); i(L))$ for all x in Σ_L^*.

We will encounter a few simple, nontrivial families of fuzzy languages in the sequel: they are the family FIN_f of finite fuzzy languages

$$\mathrm{FIN}_f = \{(\Sigma_L, \mu_L, L) \mid \Sigma_L \subset \Sigma_\omega,\ L \text{ is finite}\}\ ,$$

the family ONE_f of singleton fuzzy languages

$$\mathrm{ONE}_f = \{(\Sigma_L, \mu_L, L) \mid \Sigma_L \subset \Sigma_\omega,\ L \text{ is a singleton}\}\ ,$$

the family ALPHA_f of fuzzy alphabets

$$\mathrm{ALPHA}_f = \{(\Sigma_L, \mu_L, L) \mid \Sigma_L \subset \Sigma_\omega,\ L = \Sigma_L\}\ ,$$

and the family SYMBOL_f of singleton fuzzy alphabets

$$\mathrm{SYMBOL}_f = \{(\Sigma_L, \mu_L, L) \mid \Sigma_L \subset \Sigma_\omega,\ L = \Sigma_L\ ,\ L \text{ is a singleton }\}\ .$$

The crisp counterparts of these language families are denoted by FIN, ONE, ALPHA, and SYMBOL, respectively. Clearly, the equality $c(\mathrm{FIN}_f) = \mathrm{FIN}$ holds, as well as similar statements for the other families of languages.

Another important role will be played by the family REG_f of regular fuzzy languages, which is defined in a way very similar to its crisp counterpart REG.

Definition 3.2. Let Σ be an alphabet. The *regular fuzzy languages* over Σ are defined as follows:
(1) The fuzzy subsets $\oslash$, $\{\lambda\}$, and $\{\sigma\}$ (for each σ in Σ) of Σ^*, are regular fuzzy languages over Σ.
(2) If R_1 and R_2 are regular fuzzy languages over Σ, then so are $R_1 \cup R_2$, R_1R_2, and R_1^*.
(3) A fuzzy subset R of Σ^* is regular fuzzy language over Σ if and only if R can be obtained from the basic elements in (1) by a finite number of applications of the operations in (2).

The family of regular fuzzy languages us denoted by REG_f.

In the remainder of this paper we frequently need the concept of fuzzy substitution. It is defined in a way very similar to the notion of substitution for crisp languages; cf. [5], [6].

Definition 3.3. Let K be a family of fuzzy languages and let V be an alphabet. A mapping $\tau : V \to K$ is called a *fuzzy K-substitution* τ on V; it is extended to words over V by $\tau(\lambda) = \{\lambda\}$ with $\mu(\lambda; \tau(\lambda)) = 1$, and $\tau(\alpha_1 \ldots \alpha_n) = \tau(\alpha_1) \ldots \tau(\alpha_n)$ where $\alpha_i \in V$ $(1 \leq i \leq n)$, and to languages L over V by $\tau(L) = \bigcup\{\tau(w) \mid w \in L\}$. If for each $\alpha \in V$, $\tau(\alpha) \subseteq V^*$, then $\tau : V \to K$ is called a *fuzzy K-substitution over V*. If K equals FIN_f or REG_f, τ is called a *fuzzy finite* or a *fuzzy regular substitution*, respectively.

Given families K and K' of fuzzy languages, let $\hat{\mathrm{Sub}}(K, K') = \{\tau(L) \mid L \in K;$ τ is a fuzzy K'-substitution$\}$. A family K is *closed* under fuzzy K'-substitution if $\hat{\mathrm{Sub}}(K, K') \subseteq K$, and K is *closed under fuzzy substitution*, if K is closed under fuzzy K-substitution.

When we take K and K' equal to families of crisp languages we obtain the well-known definition of (ordinary, non-fuzzy) substitution. Therefore a ONE-substitution is just a homomorphism and an isomorphism ("renaming of symbols") is a one-to-one SYMBOL-substitution. And a fuzzy ONE_f-substitution may be called a fuzzy homomorphism.

Definition 3.4. A *fuzzy prequasoid* K is a nontrivial family of fuzzy languages that is closed under fuzzy finite substitution (i.e., $\hat{\mathrm{Sub}}(K, \mathrm{FIN}_f) \subseteq K$) and under intersection with regular fuzzy languages. A *fuzzy quasoid* is a fuzzy prequasoid that contains an infinite fuzzy language.

It is a straightforward exercise to show that each fuzzy [pre]quasoid includes the smallest fuzzy [pre]quasoid REG_f [FIN_f, respectively], whereas FIN_f is the only fuzzy prequasoid that is not a fuzzy quasoid; cf. [6].

Let $\Pi_f(K)$ denote the smallest fuzzy prequasoid that includes the family K of fuzzy languages. Similarly, let $\Phi_f(K)$ [$\Delta_f(K)$, $\Theta_f(K)$, respectively] be the smallest family of fuzzy languages that includes K and is closed under fuzzy finite substitutions [intersection with regular fuzzy languages, fuzzy homomorphisms, respectively]. Then, obviously, for each family K of fuzzy languages, we have $\Pi_f(K) = \{\Phi_f, \Delta_f, \Theta_f\}^*(K)$ or even $\Pi_f(K) = \{\Phi_f, \Delta_f\}^*(K)$. But instead of this infinite set of strings over $\{\Phi_f, \Delta_f, \Theta_f\}$ a single string suffices; viz.

Proposition 3.5. [6] *For each family K of fuzzy languages,* $\Pi_f(K) = \Theta_f \Delta_f \Phi_f(K)$.

Definition 3.6. A *full Abstract Family of Fuzzy Languages* or *full AFFL* is a nontrivial family of fuzzy languages closed under union, concatenation, Kleene $\star$, (possibly erasing) fuzzy homomorphism, inverse fuzzy homomorphism, and intersection with fuzzy regular languages. A *full substitution-closed AFFL* is a full AFFL closed under fuzzy substitution.

In many situations the following characterization of full AFFL happens to be more useful than the original definition.

Proposition 3.7. [6] *A family K of fuzzy languages is a full AFFL if and only if K is a fuzzy prequasoid closed under fuzzy regular substitution (i.e.,* $\hat{\mathrm{Sub}}(K, \mathrm{REG}_f) \subseteq K$*), and under substitution in the regular fuzzy languages (i.e.,* $\hat{\mathrm{Sub}}(\mathrm{REG}_f, K) \subseteq K$*).*

Closely related to regular fuzzy languages is a kind of fuzzy finite automaton. The next definition and equivalence result is useful, and should not come as a surprise. A proof of this characterization can be found in [6].

Definition 3.8. A *nondeterministic fuzzy finite automaton* or *NFFA* is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite fuzzy set of states, Σ is an alphabet, q_0 is an element of Q with $\mu(q_0; Q) > 0$, F is a crisp subset of the crisp set $\{q \mid \mu(q; Q) > 0\}$, and δ is a fuzzy function of type $\delta : Q \times (\Sigma \cup \{\lambda\} \to \mathcal{P}_f(Q)$. Note that M may have λ-moves.

The fuzzy function δ is extended to $\delta' : Q \times \Sigma^* \to \mathcal{P}_f(Q)$ by $\delta'(q, \lambda) = \delta(q, \lambda)$ and $\delta'(q, \sigma\omega) = \bigcup\{\delta'(q', \omega) \mid q' \in \delta(q, \sigma)\}$ for all q in Q.

The language $L(M)$ accepted by an NFFA M is defined by $\mu(x; L(M)) = \bigvee\{\mu(q; \delta'(q_0, x))) \mid q \in F\}$.

Proposition 3.9. *A fuzzy language L is regular if and only if L is accepted by a nondeterministic fuzzy finite automaton.*

4. Controlled Fuzzy Iteration Grammars

The notion of fuzzy K-iteration grammar is a straightforward modification of the definition of (ordinary) K-iteration grammar: we just replace the ordinary K-substitutions by fuzzy K-substitutions; cf. [1].

Definition 4.1. Let K be a family of fuzzy languages. A *fuzzy K-iteration grammar* G is a four-tuple $G = (V, \Sigma, U, S)$ where

- V is an alphabet (the *alphabet* of G);
- Σ is an alphabet with $\Sigma \subseteq V$ (the *terminal alphabet* of G);
- S is a symbol in V (the *initial symbol* of G);
- U is a finite set of fuzzy K-substitutions over V.

The fuzzy language $L(G)$ generated by G is defined by

$$L(G) = U^*(S) \cap \Sigma^* = \bigcup\{\tau_p(\ldots(\tau_1(S))\ldots) \mid p \geq 0;\ \ \tau_i \in U,\ \ 1 \leq i \leq p\}.$$

The family of fuzzy languages generated by fuzzy K-iteration grammars is denoted by $H_f(K)$. For each $m \geq 1$, $H_{f,m}(K)$ is the family of fuzzy languages generated by fuzzy K-iteration grammars that contain at most m fuzzy K-substitutions in U.

Definition 4.2. Let Γ be a family of crisp languages and let K be a family of fuzzy languages. A *Γ-controlled fuzzy K-iteration grammar* or *fuzzy (Γ, K)-iteration grammar* is a pair (G, M) that consists of a fuzzy K-iteration grammar $G = (V, \Sigma, U, S)$ and a *control language* M, i.e., M is a crisp language over the alphabet U. The fuzzy language $L(G, M)$ generated by (G, M) is defined by

$$L(G,M) = M(S) \cap \Sigma^* = \bigcup\{\tau_p(\ldots(\tau_1(S))\ldots) \mid p \geq 0;\ \ \tau_i \in U,\ \tau_1 \ldots \tau_p \in M\}.$$

The family of fuzzy languages generated by fuzzy (Γ, K)-iteration grammars is denoted by $H_f(\Gamma, K)$. And $H_{f,m}(\Gamma, K)$ is the family of fuzzy languages generated by fuzzy (Γ, K)-iteration grammars that contain at most m fuzzy K-substitutions in U ($m \geq 1$).

Note that in Definitions 4.1 and 4.2 $L(G)$ and $L(G, M)$, respectively, are defined in terms of union, intersection, concatenation and iterated function application for fuzzy sets; cf. Section 2 for the precise definitions of these fundamental concepts.

Clearly, we have that $H_f(K) = \bigcup\{H_{f,m}(K) \mid m \geq 1\}$ and $H_f(\Gamma, K) = \bigcup\{H_{f,m}(\Gamma, K) \mid m \geq 1\}$ for each family K of fuzzy languages and each family Γ of crisp languages.

Example 4.3. Let $\mathcal{L}$ be the type-10 lattice of Example 2.2.(3).

(1) Consider the fuzzy FIN$_f$-iteration grammar $G = (V, \Sigma, U, S)$ defined by $\Sigma = \{a, b\}$, $V = \Sigma \cup \{S\}$, and $U = \{\tau_1, \tau_2\}$ where τ_1 is an ordinary or crisp FIN-substitution with $\tau_1(S) = \{SS\}$ and $\tau_1(\alpha) = \{\alpha\}$ ($\alpha \in \Sigma$), whereas τ_2 is a FIN$_f$-substitution with $\tau_2(S) = \{a, b\}$, $\tau_2(\alpha) = \{\alpha\}$, $\mu(b; \tau_2(S)) = 0.5$ and $\mu(a; \tau_2(S)) = \mu(\alpha; \tau_2(\alpha)) = 1$ ($\alpha \in \Sigma$).

Then $L(G)$ consists of all strings w with length 2^n for some $n \geq 0$ and $\mu(w; L(G)) = 2^{-\#_b(w)}$; $\#_\sigma(x)$ denotes the number of times that the symbol σ occurs in the word x. Clearly, $c(L(G)) = \{a^{2^n} \mid n \geq 0\}$ which is the set of strings that are obtained without making any "developmental error"; cf. the discussion in Section 1. A developmental

error occurs when S changes into a b rather than into an a; the quality of the string reduces to 50% of its previous value by each such erroneous replacement.
(2) Define the REG-controlled fuzzy FIN_f-iteration grammar or $(\mathrm{REG}, \mathrm{FIN}_f)$-iteration grammar (G, M) where G is as in (1) and $M = \{\tau_1^{2k+1}\tau_2 \mid k \geq 0\}$. Now $L(G, M)$ equals the set of all strings w with length 2^n for some odd $n \geq 1$ and still we have $\mu(w; L(G, M)) = 2^{-\#_b(w)}$. Remark that $c(L(G, M)) = \{a^{2^n} \mid n \geq 0,\ \ n \text{ is odd } \}$.
(3) We modify (G, M) of (2) to a REG-controlled fuzzy REG_f-iteration grammar or $(\mathrm{REG}, \mathrm{REG}_f)$-iteration grammar (G_1, M) by redefining $\tau_2(S)$ to a REG_f-substitution with $\tau_2(S) = \{a\} \cup \{b^k \mid k \geq 1\}$, $\tau_2(\alpha) = \{\alpha\}$ for each α in Σ, $\mu(b^k; \tau_2(S)) = 2^{-k}$ for each $k \geq 1$ and $\mu(a; \tau_2(S)) = \mu(\alpha; \tau_2(\alpha)) = 1$ $(\alpha \in \Sigma)$. Then for all strings x over $\{a, b\}$, we have $\mu(x; L(G_1, M)) \geq \mu(x; L(G, M))$, $L(G, M)$ is a proper subset of $L(G_1, M)$, but $c(L(G_1, M)) = c(L(G, M))$.

Since in Example 4.3 K equals FIN_f in both (1) and (2), G may be called a fuzzy ETOL-system and (G, M) a regularly controlled fuzzy ETOL-system.

Example 4.4. By taking concrete values for the parameter K we obtain fuzzy analogues for some families of (ordinary or crisp) Lindenmayer languages; viz.

$$H_f(\mathrm{ONE}_f) = \mathrm{EDTOL}_f, \qquad H_{f,1}(\mathrm{ONE}_f) = \mathrm{EDOL}_f,$$
$$H_f(\mathrm{FIN}_f) = \mathrm{ETOL}_f, \qquad H_{f,1}(\mathrm{FIN}_f) = \mathrm{EOL}_f.$$

Readers unfamiliar with L-systems are referred to [21] for the meaning of these abbreviations.

5. Elementary Properties

In this section we establish some basic properties of Γ-controlled fuzzy K-iteration grammars and their languages that already hold under very mild restrictions on the parameters Γ and K. These results turn out to be very useful in proving more complicated and more interesting propositions to which the following two sections are devoted.

First we show that regular control does not extend the generating power of fuzzy K-iteration grammars; cf. Theorem 2.1 in [1].

Theorem 5.1. *For each family K of fuzzy languages, $H_f(\mathrm{REG}, K) = H_f(K)$ provided $K \supseteq \mathrm{ONE}$.*

Proof. Since U^* is regular for each alphabet U, the inclusion $H_f(\mathrm{REG}, K) \supseteq H_f(K)$ is obvious.

Conversely, let $(G, M) = (V, \Sigma, U, S, M)$ be an arbitrary fuzzy (REG, K)-iteration grammar where M is accepted by a complete deterministic finite automaton (Q, U, δ, q_0, Q_F) with finite set of states Q, input alphabet U, transition function $\delta : Q \times U \to Q$, initial state q_0, and set of final states Q_F.

We define a new initial symbol S_0, a set of new nonterminal symbols $N_\Sigma = \{A_a \mid a \in \Sigma\}$, and a new alphabet $V_0 = Q \cup V \cup \{S_0, F\} \cup N_\Sigma$. Define an isomorphism $\psi : V \to (V - \Sigma) \cup N_\Sigma$ by $\psi(a) = A_a$ $(a \in \Sigma)$ and $\psi(A) = A$ $(A \in V - \Sigma)$. The isomorphism ψ is extended to words and to languages in the usual way. Remember that we assumed that each family of (fuzzy) languages is closed under isomorphism.

Define the fuzzy K-iteration grammar $G_0 = (V_0, \Sigma, U_0, S_0)$ with $U_0 = \{\tau' \mid \tau \in U\} \cup \{\tau_0\}$. So for each fuzzy K-substitution τ in U there is corresponding fuzzy K-substitution τ' in U_0, defined by

$\tau'(S_0) = \{q_0S\}$,	$\mu(q_0S;\tau'(S_0)) = 1$,	
$\tau'(\alpha) = \psi(\tau(\alpha))$,		for each α in $V - \Sigma$,
$\tau(A_a) = \psi(\tau(a))$,		for each A_a in N_Σ (a in Σ),
$\tau'(q) = \{q'\}$,	$\mu(q';\tau'(q)) = 1$,	iff $\delta(q,\tau) = q'$ (q in Q),
$\tau'(\alpha) = \{F\}$,	$\mu(F;\tau'(\alpha)) = 1$,	for each α in $\Sigma \cup \{F\}$.

The additional fuzzy K-substitution τ_0 is defined as follows.

$\tau_0(q) = \{\lambda\}$,	$\mu(\lambda;\tau_0(q)) = 1$,	for each q in Q_F,
$\tau_0(q) = \{F\}$,	$\mu(F;\tau_0(q)) = 1$,	for each q in $Q - Q_F$,
$\tau_0(A_a) = \{a\}$,	$\mu(a;\tau_0(A_a)) = 1$,	for each A_a in N_Σ (a in Σ),
$\tau_0(\alpha) = \{F\}$,	$\mu(F;\tau_0(\alpha)) = 1$,	for each α in $V \cup \{S_0, F\}$.

This construction implies that for each string x in Σ^*, we have $\mu(x;L(G_0)) = \mu(x;L(G,M))$, and hence $H_f(\text{REG},K) \subseteq H_f(K)$. □

There exists a sort of reverse of Theorem 5.1 in the sense that all "productive" sequences of substitutions in a fuzzy K-iteration grammar G —i.e., those sequences that yield at least one terminal string x with $\mu(x;L(G)) > 0$— form a regular language over U; cf. Definition 5.2, Theorem 5.3 and [24].

Definition 5.2. Let $G = (V,\Sigma,U,S)$ be a fuzzy K-iteration grammar. Then the *Szilard language* of G —denoted by $Sz(G)$— is

$$Sz(G) = \{\omega \in U^* \mid \exists x \in \Sigma^* : \mu(x;\omega(S)) > 0\}.$$

The following theorem is the straightforward fuzzy counterpart of one of the main results in [24].

Theorem 5.3. *If G is a fuzzy K-iteration grammar, then its Szilard language $Sz(G)$ is a regular language.*

Proof. Let $G = (V,\Sigma,U,S)$ be a fuzzy K-iteration grammar. For each word x, we denote the set of all symbols that occur in x by $\Diamond(x)$; formally, $\Diamond(x) = \bigcap\{\Sigma \mid \Sigma \subseteq \Sigma_\omega,\ \ x \in \Sigma^*\}$.

Consider the right-linear grammar $G_0 = (V_0,U,P_0,S_0)$ where $V_0 - U = \{X \mid X \subseteq V\}$, $S_0 = \{S\}$, and P_0 is defined by

$$P_0 = \{X \to \tau Y \mid \exists x,y \in V^* : \ \Diamond(x) = X,\ \ \Diamond(y) = Y,\ \ \mu(y;\tau(x)) > 0\} \cup$$
$$\cup \{X \to \lambda \mid X \subseteq \Sigma\}.$$

Clearly, $L(G_0)$ is regular, and it is a routine matter to verify that $S_0 \Rightarrow^*_{G_0} \omega$ with $\omega \in U^*$ if and only if $\exists x \in \Sigma^* : \mu(x;\omega(S)) > 0$. □

Next we show that the number of fuzzy K-substitutions in a Γ-controlled K-iteration grammar can be reduced to two in case the parameters Γ and K satisfy some very simple conditions as in the corresponding crisp case; cf. [1].

Theorem 5.4. *Let Γ be a family of crisp languages closed under λ-free homomorphism, and let K be a family of fuzzy languages with $K \supseteq$* SYMBOL. *Then $H_{f,2}(\Gamma,K) = H_{f,m}(\Gamma,K) = H_f(\Gamma,K)$ for each $m \geq 2$.*

Proof. Of course, $H_{f,2}(\Gamma,K) \subseteq H_{f,m}(\Gamma,K) \subseteq H_f(\Gamma,K)$ holds for each $m \geq 2$. So it remains to prove that $H_f(\Gamma,K) \subseteq H_{f,2}(\Gamma,K)$.

Let $(G, M) = (V, \Sigma, U, S, M)$ be a fuzzy (Γ, K)-iteration grammar with m $(m \geq 3)$ fuzzy K-substitutions in U —say, $U = \{\tau_1, \ldots, \tau_m\}$— and let for each i $(1 \leq i \leq m)$ ψ_i be the isomorphism defined by $\psi_i(\alpha) = \alpha_i$ (α in V; each α_i is a new, unique symbol).

Construct the fuzzy (Γ, K)-iteration grammar $(G_0, M_0) = (V_0, \Sigma, U_0, S, M_0)$ with

- $V_0 = V \cup \{F\} \cup \{\psi_i(\alpha) \mid \alpha \in V,\ 1 \leq i \leq m\}$,
- $U_0 = \{\sigma_1, \sigma_2\}$ where the fuzzy K-substitutions σ_1 and σ_2 are defined respectively by

$\sigma_1(\alpha) = \{\alpha_1\}$,	$\mu(\alpha_1; \sigma_1(\alpha)) = 1$,	α in V,
$\sigma_1(\alpha_i) = \{\alpha_{i+1}\}$,	$\mu(\alpha_{i+1}; \sigma_1(\alpha_i)) = 1$,	α in V and $1 \leq i < m$,
$\sigma_1(\beta) = \{F\}$,	$\mu(F; \sigma_1(\beta)) = 1$,	β in $\{F\} \cup \{\psi_m(\alpha) \mid \alpha \in V\}$,
$\sigma_2(\alpha_i) = \tau_i(\alpha)$,		α in V and $1 \leq i \leq m$,
$\sigma_2(\beta) = \{F\}$,	$\mu(F; \sigma_2(\beta)) = 1$,	β in $V \cup \{F\}$.

- $M_0 = h(M)$ where the homomorphism $h : U^* \to U_0^*$ is defined by $h(\tau_i) = \sigma_1^i \sigma_2$ $(1 \leq i \leq m)$.

An application of τ_i of (G, M) is simulated by i times applying σ_1 (by which each α is changed into α_i) and a single application of σ_2 which carries out the actual simulation of τ_i and removes all subscripts from the symbols.

It is left to the reader to show that $\mu(x; L(G_0, M_0)) = \mu(x; L(G, M))$ for each x over Σ. Hence $H_f(\Gamma, K) \subseteq H_{f,2}(\Gamma, K)$. □

Obviously, we can combine Theorems 5.1 and 5.4 to establish a similar result for the uncontrolled case. However, we can achieve this under weaker assumptions on K by slightly modifying the proof of Theorem 5.4.

Corollary 5.5. *If K is a family of fuzzy languages with $K \supseteq$ SYMBOL, then $H_{f,2}(K) = H_{f,m}(K) = H_f(K)$ for each $m \geq 2$.*

Proof. Take M and M_0 in the proof of Theorem 5.4 equal to $M = U^*$ and $M_0 = U_0^* = \{\sigma_1, \sigma_2\}^*$, respectively. Then for each x in Σ^*, $\mu(x; L(G_0)) = \mu(x; L(G))$ holds and, consequently, $H_f(K) \subseteq H_{f,m}(K) \subseteq H_{f,2}(K)$. The converse inclusions are trivial. □

We conclude this section with a few useful inclusion properties for which we need some additional terminology.

Definition 5.6. A family Γ of crisp languages is closed under *left marking* [*right marking*] if for each language L in Γ with $L \subseteq \Sigma^*$ for some Σ, and for each symbol c not in Σ, the language $\{c\}L$ [$L\{c\}$, respectively] belongs to Γ. And Γ is closed under *full marking* if Γ is closed under both left and right marking. Frequently, we write cL and Lc rather than $\{c\}L$ and $L\{c\}$, respectively.

Proposition 5.7. *(1) Let Γ be a family of crisp languages closed under right marking, and let K be a family of fuzzy languages with $K \supseteq$ ONE. Then the inclusions $\Gamma \subseteq H_f(\Gamma, K)$ and $K \subseteq H_f(\Gamma, K)$ hold.*
(2) Let Γ be a family of crisp languages closed under (i) left or right marking, (ii) union or concatenation, and (iii) Kleene star. If K is a family of fuzzy languages with $K \supseteq$ SYMBOL, then $H_f(K) \subseteq H_f(\Gamma, K)$.

Proof. (1) Consider an arbitrary crisp language L_0 over U_0 in the family Γ. Define the fuzzy (Γ, K)-iteration grammar $(G, M) = (V, U_0, U, S, M)$ with $U = U_0 \cup \{\sigma\}$, $M = L_0\sigma$, and U consists of fuzzy K-substitutions defined by

$$\begin{array}{ll} \tau(S) = \{\tau S\}, & \tau \in U_0, \\ \tau(\alpha) = \{\alpha\}, & \alpha \in U_0,\ \tau \in U_0, \\ \sigma(S) = \{\lambda\}, & \\ \sigma(\alpha) = \{\alpha\}, & \alpha \in U_0. \end{array}$$

All degrees of membership are equal to 1 (or to 0 in all other, unmentioned cases). So (G, M) is actually a crisp (Γ, K)-iteration grammar with $L(G, M) = L_0$. Consequently, we have $\Gamma \subseteq H_f(\Gamma, K)$.

Similarly, let L_0 be a fuzzy language over Σ and let M_0 be an arbitrary nonempty crisp language over U_0. We define the fuzzy (Γ, K)-iteration grammar $(G, M) = (V, \Sigma, U, S, M)$ where $V = \Sigma \cup \{S\}$, $U = U_0 \cup \{\sigma\}$ $(\sigma \notin U_0)$, $M = M_0\sigma$, and the fuzzy K-substitutions are defined by

$$\begin{array}{lll} \tau(\alpha) = \{\alpha\}, & \mu(\alpha; \tau(\alpha)) = 1, & \alpha \in V,\ \tau \in U, \\ \sigma(S) = L_0, & \mu(x; \sigma(S)) = \mu(x; L_0), & \text{for all } x \text{ over } \Sigma, \\ \sigma(\alpha) = \{\alpha\}, & \mu(\alpha; \sigma(\alpha)) = 1, & \alpha \in \Sigma. \end{array}$$

Then $\mu(x; L(G, M)) = \mu(x; L_0)$ for all x over Σ, and thus $K \subseteq H_f(\Gamma, K)$.

(2) Let $G = (V, \Sigma, U, S)$ be an arbitrary fuzzy K-iteration grammar with $U = \{\tau_1, \ldots, \tau_n\}$ and let M_0 be a nonempty crisp language over U_0 from Γ such that $U \cap U_0 = \oslash$. If the family Γ is closed under union [concatenation], then the crisp language $M = (M_0\tau_1 \cup M_0\tau_2 \cup \ldots \cup M_0\tau_n)^\star$ [or $M = ((M_0\tau_1)^\star(M_0\tau_2)^\star \ldots (M_0\tau_n)^\star)^\star$, respectively] is also in Γ.

Finally, we define the fuzzy (Γ, K)-iteration grammar (G_1, M) by $(G_1, M) = (V, \Sigma, U_1, S, M)$ with $U_1 = U \cup U_0$ and for each τ in U_0 and for each α in V, $\tau(\alpha) = \{\alpha\}$ with $\mu(\alpha; \tau(\alpha)) = 1$. Then $\mu(x; L(G_1, M)) = \mu(x; L(G))$ for each x over Σ and, consequently, $H_f(K) \subseteq H_f(\Gamma, K)$. □

6. The Main Results

In Section 1 we argued that in order to model developmental errors we should allow a countable rather than a finite number of productions in each table (or substitution). This resulted in the notion of Γ-controlled fuzzy K-iteration grammar and the corresponding language family $H_f(\Gamma, K)$.

In this section we address the question to which extend we can enlarge the family K of fuzzy languages and still remain within the family $H_f(\Gamma, K)$. The answer (Theorem 6.1 and Corollaries 6.2, 6.3 and 6.4)) is rather surprising and implies that both families $H_f(\Gamma, K)$ and $H_f(K)$ possess very strong closure properties; this latter subject will be discussed in Section 7.

For families Γ_1 and Γ_2 of crisp languages, $\hat{\text{Sub}}(\Gamma_1, \Gamma_2)$ denotes the family of crisp languages that results from substituting Γ_2-languages into Γ_1-languages, i.e., $\hat{\text{Sub}}(\Gamma_1, \Gamma_2) = \{\tau(L) \mid L \in \Gamma_1,\ \tau \text{ is a } \Gamma_2\text{-substitution}\}$. A family Γ is *closed under substitution* if $\hat{\text{Sub}}(\Gamma, \Gamma) \subseteq \Gamma$. Of course, these concepts are well-known special instances of Definition 3.3.

Theorem 6.1. *Let Γ_1 and Γ_2 be families of crisp languages and let Γ_2 be closed under full marking, union or concatenation, and Kleene $\star$. If K is a family of fuzzy languages with $K \supseteq$ ALPHA, then $H_f(\Gamma_1, H_f(\Gamma_2, K)) \subseteq H_f(\hat{\text{Sub}}(\Gamma_1, \Gamma_2), K)$.*

Proof. Consider an arbitrary Γ_1-controlled fuzzy $H_f(\Gamma_2, K)$-iteration grammar $(G, M) = (V, \Sigma, U, S, M)$, where each τ in U is a fuzzy $H_f(\Gamma_2, K)$-substitution over V. For each such fuzzy $H_f(\Gamma_2, K)$-substitution τ in U and each symbol α in V, we assume that $\mu(x; \tau(\alpha)) = \mu(x; L(G_{\tau\alpha}, M_{\tau\alpha}))$ holds for each x over V. Here $(G_{\tau\alpha}, M_{\tau\alpha}) = (V_{\tau\alpha}, V, U_{\tau\alpha}, S_{\tau\alpha}, M_{\tau\alpha})$ ($\tau \in U$ and $\alpha \in V$) are fuzzy (Γ_2, K)-iteration grammars that have mutually disjoint nonterminal alphabets $V_{\tau\alpha} - V$ as well as mutually disjoint sets of fuzzy K-substitution names $U_{\tau\alpha}$.

We also assume that the fuzzy (Γ_2, K)-iteration grammars $(G_{\tau\alpha}, M_{\tau\alpha})$ meet the following conditions: (i) for each a in V and each σ in $U_{\tau\alpha}$: $\sigma(a) = \{a\}$ with $\mu(a; \sigma(a)) = 1$, and (ii) if an intermediate string ω in a derivation due to $(G_{\tau\alpha}, M_{\tau\alpha})$ contains a symbol of V, then for each σ in $U_{\tau\alpha}$: $\sigma(\omega) = \{\omega\}$, while for all u over $U_{\tau\alpha}$ and each w over $V_{\tau\alpha}$, we have $\mu(\omega; \sigma u(w)) = \mu(\omega; u(w))$. Otherwise, we introduce for each a in V a new nonterminal symbol A_a and we replace each occurrence of a in $(G_{\tau\alpha}, M_{\tau\alpha})$ by A_a. Each fuzzy substitution is extended with $\sigma(\beta) = \{\beta\}$, $\mu(\beta; \sigma(\beta)) = 1$ with $\beta \in V \cup \{F_0\}$, where F_0 is a new rejection symbol. Finally, we add a new fuzzy substitution φ defined by

$$\begin{array}{lll} \varphi(A_a) = \{a\}, & \mu(a; \varphi(A_a)) = 1, & \text{for each } a \text{ in } V, \\ \varphi(a) = \{a\}, & \mu(a; \varphi(a)) = 1, & \text{for each } a \text{ in } V \cup \{F_0\}, \\ \varphi(\beta) = \{F_0\}, & \mu(F_0; \varphi(\beta)) = 1, & \text{for each } \beta \text{ in } V_{\tau\alpha} - V, \end{array}$$

and we replace the control language $M_{\tau\alpha}$ by $M_{\tau\alpha}\varphi$.

In order to show that the fuzzy language $L(G, M)$ belongs to the family $H_f(\hat{\text{Sub}}(\Gamma_1, \Gamma_2), K)$, we construct a fuzzy $(\hat{\text{Sub}}(\Gamma_1, \Gamma_2), K)$-iteration grammar $(G_0, M_0) = (V_0, \Sigma, U_0, S, M_0)$ such that $\mu(x; L(G_0, M_0)) = \mu(x; L(G, M))$ holds for each x in Σ^*. The definition of (G_0, M_0) is as follows.

- $V_0 = \bigcup_{\tau,\alpha}(V_{\tau\alpha} \cup \{S'_{\tau\alpha}\}) \cup \{F\}$ where F is a rejection symbol and each $S'_{\tau\alpha}$ is a new nonterminal symbol associated with $S_{\tau\alpha}$. Remark that $S \in V$, and since $V \subseteq V_{\tau\alpha} \subseteq V_0$, we have $S \in V_0$.
- $U_0 = \{\sigma_0\} \cup \{\sigma_\tau \mid \tau \in U\} \cup \{\sigma'_{\tau\alpha k} \mid \sigma_{\tau\alpha k} \in U_{\tau\alpha}\}$.
- The fuzzy K-substitutions in U_0 are defined in the following way:

(a) For the initial fuzzy K-substitution σ_0 we have with degree of membership equal to 1 in all the following instances:

$$\begin{array}{ll} \sigma_0(\alpha) = \{S'_{\tau\alpha} \mid \tau \in U\}, & \alpha \in V, \\ \sigma_0(\alpha) = \{F\}, & \alpha \notin V. \end{array}$$

(b) For each τ in U the fuzzy K-substitution σ_τ is defined by

$$\begin{array}{ll} \sigma_\tau(S'_{\tau\alpha}) = \{S'_{\tau\alpha}, S_{\tau\alpha}\}, & \alpha \in V, \\ \sigma_\tau(\alpha) = \{\alpha\}, & \alpha \in V, \\ \sigma_\tau(\alpha) = \{F\}, & \alpha \notin V \cup \{S'_{\tau\alpha}\}, \end{array}$$

where all degrees of membership are again equal to 1.

(c) For each fuzzy K-substitution $\sigma_{\tau\alpha k}$ from $U_{\tau\alpha}$ we define a corresponding fuzzy K-substitution $\sigma'_{\tau\alpha k}$ by

$$\begin{array}{lll} \sigma'_{\tau\alpha k}(\beta) = \sigma_{\tau\alpha k}(\beta), & & \beta \in V_{\tau\alpha}, \\ \sigma'_{\tau\alpha k}(S'_{\tau\beta}) = \{S'_{\tau\beta}\}, & \mu(S'_{\tau\beta}; \sigma'_{\tau\alpha k}(S'_{\tau\beta})) = 1, & \beta \in V, \\ \sigma'_{\tau\alpha k}(\beta) = \{F\}, & \mu(F; \sigma_{\tau\alpha k}(\beta)) = 1, & \text{otherwise.} \end{array}$$

• The control language M_0 is defined by $M_0 = \gamma(M)$ where γ is the Γ_2-substitution defined by

$$\gamma(\tau) = M_\tau, \qquad \tau \in U,$$

where the languages M_τ with $\tau \in U$ satisfy —assuming $V = \{\alpha_1, \ldots, \alpha_n\}$—

$$\begin{aligned} M_\tau &= \sigma_0(\sigma_\tau M_{\tau\alpha_1} \cup \ldots \cup \sigma_\tau M_{\tau\alpha_n})^*, && \text{if } \Gamma_2 \text{ is closed under union, and} \\ M_\tau &= \sigma_0(\sigma_\tau M_{\tau\alpha_1} \ldots \sigma_\tau M_{\tau\alpha_n})^*, && \text{if } \Gamma_2 \text{ is closed under concatenation.} \end{aligned}$$

Clearly, each language M_τ ($\tau \in U$) belongs to the family Γ_2.

Each step in any derivation according to the Γ_1-controlled fuzzy (Γ_2, K)-iteration grammar (G, M) is simulated by a finite number of derivational steps of the fuzzy $(\text{Sûb}(\Gamma_1, \Gamma_2), K)$-iteration grammar (G_0, M_0) in the following way.

For each intermediate string in a derivation of (G, M) there is an identical string over V in the simulation by (G_0, M_0). However, going from such a string to the next one over V —i.e., the actual simulation of the application of a fuzzy (Γ_2, K)-substitution τ from U in a (G, M)-derivation— takes a finite number of steps controlled by the language M_τ. So the simulation of a single step according to τ by M_τ proceeds as follows. First, all symbols α from V are converted into $S'_{\tau\alpha}$ by a single application of σ_0. Next an application of σ_τ checks whether all first indices of these primed initial symbols are indeed equal to τ, otherwise at least one occurrence of the rejection symbol F is introduced. Simultaneously, some of the occurrences of the primed initial symbols $S'_{\tau\alpha}$ may be changed into their unprimed counterparts $S_{\tau\alpha}$. And symbols from $\bigcup_{\rho,\beta} V_{\rho\beta} - V$ are rewritten into the rejection symbol F. Obviously, the unprimed symbols $S_{\tau\alpha}$ start an actual derivation according to $(G_{\tau\alpha}, M_{\tau\alpha})$, i.e., according to the fuzzy K-substitutions $\sigma'_{\tau\alpha k}$ due to the control language $M_{\tau\alpha}$. Clearly, the definitions of M_τ and of σ_τ allow different occurrences of $S_{\tau\alpha}$ be rewritten under different control words from $M_{\tau\alpha}$. Finally, after the simulation of a τ-step only occurrences of symbols from V will survive the simulation of a subsequent τ'-step and contribute to the derivation of a possible terminal substring in the end.

By a long, straightforward correctness proof —which we leave to the interested reader— one can establish that for each string x over Σ, we have $\mu(x; L(G_0, M_0)) = \mu(x; L(G, M))$, and, consequently, we have established the inclusion $H_f(\Gamma_1, H_f(\Gamma_2, K)) \subseteq H_f(\text{Sûb}(\Gamma_1, \Gamma_2), K)$. □

Corollary 6.2. *(1) Let Γ be a family of crisp languages closed under full marking and under substitution that satisfies $\Gamma \supseteq$ REG. If K is a family of fuzzy languages with $K \supseteq$ ALPHA $\cup$ ONE, then $H_f(\Gamma, H_f(\Gamma, K)) = H_f(\Gamma, K)$.*
(2) Let Γ be a family of crisp languages that is closed under full marking, union, concatenation, and Kleene $\star$. If K is a family of fuzzy languages with $K \supseteq$ ALPHA $\cup$ ONE, then $H_f(H_f(\Gamma, K)) = H_f(\Gamma, K)$.

Proof. (1) follows from Theorem 6.1 in which we take $\Gamma_1 = \Gamma_2 = \Gamma$, Proposition 5.7.(1), and the fact that a family of crisp languages is closed under union, concatenation, and Kleene $\star$ if and only if it is closed under substitution into the regular languages (Proposition 3.3.1 in [9]).

(2) is implied by (i) Theorem 6.1 (where we take Γ_1 and Γ_2 equal to REG and Γ, respectively), (ii) Theorem 5.1, (iii) Proposition 3.3.1 in [9] (as in the proof of

6.2.(1)), and finally (iv) the inclusion $H_f(\Gamma, K) \subseteq H_f(H_f(\Gamma, K))$ due to Proposition 5.7(1). □

Corollary 6.3. *If K is a family of fuzzy languages with $K \supseteq$ ALPHA $\cup$ ONE, then $H_f(H_f(K)) = H_f(K)$.*

Proof. If we take Γ equal to REG, then the result follows from Theorem 5.1 and Corollary 6.2.(2) immediately. □

Corollary 6.4. ETOL$_f$ = H_f(ETOL$_f$) = $H_f(H_f$(FIN$_f$)) = H_f(FIN$_f$).

Proof. Example 4.4 and Corollary 6.3 with K equal to FIN$_f$. □

This latter corollary shows that, in order to stay within the framework of ETOL$_f$-languages (i.e., H_f(FIN$_f$)-languages; cf. Example 4.4.), we have to restrict the infinite fuzzy sets $\tau(\alpha)$ consisting of developmental rules together with developmental errors to ETOL$_f$-languages as H_f(ETOL$_f$) $\subseteq$ ETOL$_f$; cf. the discussion in Section 1. Of course, a similar remark applies in the more general case (Corollary 6.3) but the extension from finite sets to countably infinite fuzzy sets is a more striking phenomenon.

7. Closure Properties

We already remarked that Theorem 6.1 and its corollaries imply that the families $H_f(\Gamma, K)$ and $H_f(K)$ of fuzzy languages possess very strong closure properties under minor assumptions and the families Γ and K. In this section we first consider some simple closure properties (Lemmas 7.1 and 7.2) before we consider the more important ones (Theorem 7.5) due to our results from Section 6.

Lemma 7.1. *Let K be a family of fuzzy languages with $K \supseteq$ FIN$_f$, and let Γ be a family of crisp languages closed under right marking. Then the families of fuzzy languages $H_f(K)$ and $H_f(\Gamma, K)$ are closed under fuzzy finite substitution.*

Proof. Let $G = (V, \Sigma, U, S)$ be a fuzzy K-iteration grammar and let $\sigma : \Sigma \to \Delta^*$ be a fuzzy finite substitution. Without loss of generality we assume that the alphabets Σ and Δ are disjunct.

Consider the fuzzy K-iteration grammar $G_0 = (V_0, \Delta, U_0, S)$ where $V_0 = V \cup \Delta \cup \{F\}$, $U_0 = \{\tau' \mid \tau \in U\} \cup \{\sigma'\}$ with

$$\begin{aligned} &\sigma'(\alpha) = \sigma(\alpha), && && \alpha \in \Sigma, \\ &\sigma'(\alpha) = \{F\}, && \mu(F; \sigma'(\alpha)) = 1, && \alpha \notin \Sigma, \end{aligned}$$

and for each τ in U we define

$$\begin{aligned} &\tau'(\alpha) = \tau(\alpha), && && \alpha \in V, \\ &\tau'(\alpha) = \{F\}, && \mu(F; \tau'(\alpha)) = 1, && \alpha \in \Delta \cup \{F\}. \end{aligned}$$

Then for each string x over Δ, we have $\mu(x; \sigma(L(G))) = \mu(x; L(G_0))$.

In the Γ-controlled case we depart from (G, M) and we construct (G_0, M_0) with G_0 as above and $M_0 = \varphi(M)\{\sigma'\}$ where φ is the isomorphism that maps each τ on τ'. □

Lemma 7.2. *Let K be a fuzzy prequasoid, and let Γ be a family of crisp languages closed under full marking. Then the familes of fuzzy languages $H_f(K)$ and $H_f(\Gamma, K)$ are closed under intersection with regular fuzzy languages.*

Proof. Let $G = (V, \Sigma, U, S)$ be a fuzzy K-iteration grammar, and let R be a regular fuzzy language accepted by a nondeterministic fuzzy finite automaton $(Q, \Sigma, \delta, q_0, F)$; cf. Proposition 3.9.

Consider the fuzzy K-iteration grammar $G_0 = (V_0, \Sigma, U_0, S_0)$ where $V_0 = \Sigma \cup \{S_0, F\} \cup \{[q, \alpha, q'] \mid q, q' \in Q,\ \alpha \in V\}$, $U_0 = \{\sigma_0, \sigma_1\} \cup \{\tau' \mid \tau \in U\}$, with

$$\begin{aligned} \sigma_0(S_0) &= \{[q_0, S, q] \mid q \in F\}, && q \in F, \\ \sigma_0(\alpha) &= \{\alpha\}, && \alpha \in V_0 - \{S_0\}, \\ \sigma_1(\alpha) &= \{\alpha\}, && \alpha \in \Delta \cup \{S_0, F\}; \end{aligned}$$

the degrees of membership are equal to 1 for all these instances. But for

$$\sigma_1([q, \alpha, q']) = \{\alpha \mid q' \in \delta(q, \alpha)\} \cup \{F\}, \qquad \alpha \in V,\ q, q' \in Q,$$

we have $\mu(\alpha; \sigma_1([q, \alpha, q'])) = \mu(q'; \delta(q, \alpha))$ and $\mu(F; \sigma_1([q, \alpha, q']) = 1$.

For each τ in U, we define the fuzzy substitution τ' over V_0 by

$$\begin{aligned} \tau'([q, \alpha, q']) = {} & \{[q, \alpha_1, q_1][q_1, \alpha_2, q_2] \ldots [q_{n-1}, \alpha_n, q'] \mid q_1, \ldots, q_{n-1} \in Q; \\ & \alpha_1 \alpha_2 \ldots \alpha_n \in \tau(\alpha),\ n \geq 1\} \cup E((\tau, \alpha, q, q'), \quad \alpha \in V,\ q, q' \in Q, \end{aligned}$$

with $E((\tau, \alpha, q, q') = \textbf{if } \lambda \in \tau(\alpha) \text{ and } q = q' \textbf{ then } \{\lambda\} \textbf{ else } \{F\}$. For the degrees of membership we have

$$\begin{aligned} & \mu([q, \alpha_1, q_1] \ldots [q_{n-1}, \alpha_n, q']; \tau'([q, \alpha, q'])) = \mu(\alpha_1 \ldots \alpha_n; \tau(\alpha)), \qquad n \geq 1, \\ & \mu(\lambda; \tau'([q, \alpha, q'])) = \textbf{if } \lambda \in \tau(\alpha) \text{ and } q = q' \textbf{ then } \mu(\lambda; \tau(\alpha)) \textbf{ else } \mathbf{0}, \\ & \mu(F; \tau'([q, \alpha, q'])) = 1. \end{aligned}$$

Since K is a fuzzy prequasoid, it easy to show that each τ' is a fuzzy K-substitution over V_0. The proof that for each string x over Σ, $\mu(x; L(G_0)) = \mu(x; L(G) \cap R)$ holds is also left to the reader.

When G is provided with a crisp control language M from the family Γ, we construct (G_0, M_0) with $M_0 = \{\sigma_0\}\varphi(M)\{\sigma_1\}$, where φ is as in the proof of Lemma 7.1. □

We now turn to more complicated closure properties for fuzzy languages.

Definition 7.3. A family K of fuzzy languages is closed under *iterated fuzzy substitution* if for each fuzzy language L in K over some alphabet V ($L \subseteq V^*$), and each finite set U of fuzzy K-substitutions over V, the language $U^*(L)$ defined by

$$U^*(L) = \bigcup\{\tau_p(\ldots(\tau_1(L))\ldots) \mid p \geq 0;\ \ \tau_i \in U,\ \ 1 \leq i \leq p\}$$

belongs to K.

A *hyper-algebraically closed full Abstract Family of Fuzzy Languages*, or *full hyper-AFFL* for short, is a full AFFL closed under nested iterated fuzzy substitution.

For a fuzzy prequasoid closure under iterated fuzzy substitution implies closure under many of the operations related to the notion of full AFFL; using Proposition 3.7, Definitions 7.3 and 3.6 it is straightforward to establish the following characterization.

Proposition 7.4. *A family K of fuzzy languages is a full hyper-AFFL if and only if K is a fuzzy prequasoid and $H_f(K) = K$.*

Each full hyper-AFFL is a full super-AFFL (i.e., a full AFFL closed under iterated nested fuzzy substitution; a substitution τ is *nested* if $\alpha \in \tau(\alpha)$ holds for each symbol α.), and each full super AFFL is in its turn a full substitution-closed AFFL [5], but none of the converse implications holds.

Now we are ready for the main results of this section.

Theorem 7.5. *If K is a fuzzy prequasoid and if Γ is a family of crisp languages closed under full marking, union, concatenation, and Kleene $\star$, then the family of fuzzy languages $H_f(\Gamma, K)$ is a full hyper-AFFL.*

Proof. By Lemmas 7.1 and 7.2 we obtain the fact that $H_f(\Gamma, K)$ is a fuzzy prequasoid. Then by Proposition 7.4 and Corollary 6.2.(2) the result follows. □

Theorem 7.6. *(1) If K is a fuzzy prequasoid, then $H_f(K)$ is a full hyper-AFFL.*
(2) For each arbitrary family K of fuzzy languages, $H_f\Pi_f(K)$ is the smallest full hyper-AFFL that includes K.
(3) For each arbitrary family K of fuzzy languages, $H_f\Theta_f\Delta_f\Phi_f(K)$ is the smallest full hyper-AFFL that includes K.

Proof. (1) The statement follows immediately from Lemmas 7.1 and 7.2 together with Corollary 6.3.

(2) Let $\hat{\mathcal{H}}_f(K)$ be the smallest full hyper-AFFL that includes K. By the inclusion $K \subseteq \hat{\mathcal{H}}_f(K)$ and the monotonicity of both H_f and Π_f, we have $H_f\Pi_f(K) \subseteq H_f\Pi_f\hat{\mathcal{H}}_f(K)$. According to Proposition 7.4 this yields $H_f\Pi_f(K) \subseteq \hat{\mathcal{H}}_f(K)$. Now Theorem 7.6.(1) implies that $H_f\Pi_f(K)$ is a full hyper-AFFL that includes K. Hence we obtain that $\hat{\mathcal{H}}_f(K) = H_f\Pi_f(K)$.

(3) By Theorem 7.6.(2) and Proposition 3.5. □

By Proposition 7.4 we have that a family of fuzzy languages K is a full hyper-AFFL if and only if $\Pi_f(K) = K$ and $H_f(K) = K$. Consequently, the smallest full hyper-AFFL $\hat{\mathcal{H}}_f(K)$, that includes a family K of fuzzy languages, equals $\hat{\mathcal{H}}_f(K) = \bigcup\{w(K) \mid w \in \{\Pi_f, H_f\}^*\}$ or, written equivalently, $\hat{\mathcal{H}}_f(K) = \{\Pi_f, H_f\}^*(K)$. According Theorem 7.6.(2) this infinite set of strings over the alphabet $\{\Pi_f, H_f\}$ can be reduced to the single string $H_f\Pi_f$. Of course, a similar remark applies to Theorem 7.6.(3).

From the fact that FIN_f is the smallest fuzzy prequasoid, Theorem 7.6.(1), Corollary 6.4, Example 4.4, and the monotonicity of the operator H_f we obtain

Corollary 7.7. ETOL_f *is the smallest full hyper-AFFL.*

8. Concluding Remarks

In the previous sections we extended the concept of Γ-controlled K-iteration grammar from [1] to its fuzzy analogue in order to model the phenomenon of "developmental error". Many of the results that we have established are straightforward generalizations of similar statements for the crisp case from [1], [24] once the language-theoretic operations —like homomorphism, substitution and concatenation— are extended in the right way for fuzzy languages; cf. Section 2. On the other hand non-fuzzy versions of Theorem 6.1 and Corollary 6.2.(1) are proper generalizations of the main result in [1] which is more or less equivalent to the crisp counterpart of Corollary 6.2.(2).

Obviously, all our results apply to fuzzy ETOL languages as well; they are obtained by taking the parameter family K of fuzzy languages equal to the family FIN_f of finite fuzzy languages. The precise formulation of these statements for Γ-controlled ETOL_f-languages are left to the interested reader.

In the definition of fuzzy K-iteration grammar each element in U is an arbitrary fuzzy K-substitution over V. Restricting each τ in U to a nested fuzzy K-substitution —i.e., $\mu(\alpha;\tau(\alpha)) = 1$ for each $\alpha \in V$— results in the concept of fuzzy context-free K-grammar; cf. [3], [4]. A further restriction to not-self-embedding nested fuzzy K-substitutions yields the notion of fuzzy regular K-grammar; cf. [5]. Both types of grammars have properties rather similar than those presented in this paper. Particularly with respect to closure properties there are many similarities and the question arises whether a uniform approach as the one in [2] for crisp languages is also possible for families of fuzzy languages. On the other hand there are some differences between fuzzy regular or context-free K-grammars and fuzzy K-iteration grammars. E.g., for fuzzy regular and fuzzy context-free K-grammars we can reduce the number of substitutions to 1 rather than to 2 (cf. Theorem 5.4), which implies that providing these grammars with a control language is probably not very challenging.

Next we return to a few matters discussed in Section 1. First, we want to reconsider the effect of developmental errors on the quality of the filament. In Section 1 we argued that each developmental error should properly change this quality, and therefore the underlying lattice-ordered structure $\mathcal{L}$ should possess an infinite number of elements. Clearly, the real closed interval $[0,1]$ —even restricted to its computable or rational elements; cf. [8]— satisfies this condition, which is one reason for its popularity. But other instances of $\mathcal{L}$ may be useful too. E.g. in case we want to count symbols, i.e. to count cell states in filaments, the elements of $\mathcal{L}$ may be Parikh-vectors with $0 = [0,0,\ldots,0]$, and $1 = [\infty,\infty,\ldots,\infty]$ as smallest and largest element in $\mathcal{L}$. Note that $\mathcal{L}$ has countably infinite elements too in this example.

Two examples of biologically motivated control languages have been mentioned in Section 1: the sequence of days and nights, and the sequence of seasons. Both sets of sequences are regular languages. So the obvious question is: are there any non-regular events in biology/nature? Other sets of sequences —like the proper order of the days in a week, of the months in a year— are unsuitable candidates: apart from being regular sets, they are also human artifacts rather than natural or biological events.

References

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

2. Asveld, P.R.J.: An algebraic approach to incomparable families of formal languages, pp. 455–475 in [22].

3. Asveld, P.R.J.: Towards robustness in parsing — Fuzzifying context-free language recognition, pp. 443–453 in: J. Dassow, G. Rozenberg & A. Salomaa (eds.): *Developments in Language Theory II – At the Crossroads of Mathematics, Computer Science and Biology* (1996), World Scientific, Singapore.

4. Asveld, P.R.J.: A fuzzy approach to erroneous inputs in context-free language recognition, pp. 14–25 in: *Proc. 4th Internat. Workshop on Parsing Technologies* (1995), Prague/Karlovy Vary, Czech Republic.

5. Asveld, P.R.J.: The non-self-embedding property for generalized fuzzy context-free grammars, Memorandum Informatica 96-08 (1996), Dept. of Comp. Sci., Twente University of Technology, Enschede, the Netherlands. Presented at *8th Internat. Conf. on Automata and Formal Languages* (1996), Salgótarján, Hungary.

6. Asveld, P.R.J.: A note on Full Abstract Families of Fuzzy Languages (Full AFFL). In preparation.

7. Dassow, J. & Păun, G.: *Regulated Rewriting in Formal Language Theory* (1989), Springer-Verlag, Berlin, etc.

8. Gerla, G.: Fuzzy grammars and recursively enumerable fuzzy languages, *Inform. Sci.* **60** (1992) 137–143.

9. Ginsburg, S.: *Algebraic and Automata-Theoretic Properties of Formal Languages* (1975), North-Holland, Amsterdam.

10. Ginsburg, S. & Rozenberg, G.: TOL schemes and control sets, *Inform. Contr.* **27** (1975) 109–125.

11. Goguen, J.A.: L-fuzzy sets, *J. Math. Analysis Appl.* **18** (1967) 145–174.

12. Harrison, M.A.: *Introduction to Formal Language Theory* (1978), Addison-Wesley, Reading, Mass.

13. Herman, G.T. & Rozenberg, G.: *Developmental Systems and Languages* (1975), North-Holland, Amsterdam.

14. Hopcroft, J.E. & Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation* (1979), Addison-Wesley, Reading, Mass.

15. Lindenmayer, A.: Mathematical models for cellular interactions in development, Parts I and II, *J. Theor. Biology* **18** (1968) 280–315.

16. Lindenmayer, A.: Developmental systems without cellular interaction, their languages and grammars, *J. Theor. Biology* **30** (1971) 455–484.

17. Lee, E.T. & Zadeh, L.A.: Note on fuzzy languages, *Inform. Sci.* **1** (1969) 421–434.

18. Nielsen, M.: EOL systems with control devices, *Acta Inform.* **4** (1975) 373–386.

19. Prasad, N., Mahajan, M. & Krithivasan, K.: Fuzzy L-systems, *Internat. J. Comput. Math.* **36** (1990) 139–161.

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

21. Rozenberg, G. & Salomaa, A.: *The Mathematical Theory of L Systems* (1980), Academic Press, New York.

22. Rozenberg, G. & Salomaa, A. (eds.): *Lindenmayer Systems — Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology* (1992), Springer-Verlag, Berlin, etc.

23. Salomaa, A.: *Formal Languages* (1973), Academic Press, New York.

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