A FUZZY APPROACH TO ERRONEOUS INPUTS IN CONTEXT-FREE LANGUAGE RECOGNITION

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 – Using fuzzy context-free grammars one can easily describe a finite number of ways to derive incorrect strings together with their degree of correctness. However, in general there is an infinite number of ways to perform a certain task wrongly. In this paper we introduce a generalization of fuzzy context-free grammars, the so-called fuzzy context-free K-grammars, to model the situation of making a finite choice out of an infinity of possible grammatical errors during each context-free derivation step. Under minor assumptions on the parameter K this model happens to be a very general framework to describe correctly as well as erroneously derived sentences by a single generating mechanism.

Our first result characterizes the generating capacity of these fuzzy context-free K-grammars. As consequences we obtain: (i) bounds on modeling grammatical errors within the framework of fuzzy context-free grammars, and (ii) the fact that the family of languages generated by fuzzy context-free K-grammars shares closure properties very similar to those of the family of ordinary context-free languages.

The second part of the paper is devoted to a few algorithms to recognize fuzzy context-free languages: viz. a variant of a functional version of Cocke–Younger–Kasami's algorithm and some recursive descent algorithms. These algorithms turn out to be robust in some very elementary sense and they can easily be extended to corresponding parsing algorithms.

1. Introduction

When we say that a parser is robust it is not quite clear what we mean, since the notion of robustness reflects in fact an informal collection of aspects related to the improper use or the exceptional behavior of the parser. One aspect that is mentioned frequently in this context, concerns the adequate behavior of the parser to small errors in its input. To this aspect and, particularly, the formal distinction between small and big errors, their arising in the derivational process due to a context-free grammar as well as their treatment in the corresponding recognition process, the present paper is devoted.

The first problem that we encounter, is the distinction between small errors ("tiny mistakes") and big errors ("capital blunders") in the input of a parser or recognizer for a context-free language. In traditional formal language theory there is no possibility for such a subtle distinction. Indeed, given a language L_0 over an alphabet Σ and a string x over Σ, then x is either *in* or *out* the language L_0. This dichotomy of the set Σ^* of strings over Σ is also apparent when we look at the membership or characteristic function $\mu : \Sigma^* \to \{0, 1\}$ of the set L_0 which is defined by $\mu(x;L_0) = 1$ if and only if $x \in L_0$ and $\mu(x;L_0) = 0$ if and only if $x \notin L_0$. But now the notion of fuzzy set may solve this problem, since a fuzzy language over Σ^* is defined in terms of a membership function $\mu : \Sigma^* \to [0, 1]$. Note that the two-element set $\{0, 1\}$ has been replaced by the continuous interval $[0, 1]$ and $\mu(x;L_0)$ expresses the degree of membership of the element x with respect to the language L_0. Thus x may fully belong to L_0 (when $\mu(x;L_0) = 1$), completely be out of L_0 (when $\mu(x;L_0) = 0$), or anything in between. In case we choose two appropriate constants δ and Δ with $0 < \delta, \Delta < ½$ we are able to distinguish "tiny mistakes" (those strings x over Σ satisfying $1 - \delta \leq \mu(x;L_0) < 1$) from "capital blunders" (strings with $0 < \mu(x;L_0) \leq \Delta$).

The next matter we discuss is: how do errors show up, and which errors (small or big) do we consider. Henceforth, we assume that our language L_0 is generated by a context-free grammar $G = (V, \Sigma, P, S)$ consisting of an alphabet V, a terminal alphabet Σ ($\Sigma \subseteq V$), an initial symbol S ($S \in V$), and a finite set P of rules ($P \subseteq N \times V^*$ where $N = V - \Sigma$). However, it turns out to be more convenient to view P as a function from V to finite subsets of V^* (i.e., finite languages over V) rather than as a subset of $N \times V^*$. To be more specific, let A be an arbitrary nonterminal with rules $A \to \omega_1 \mid \omega_2 \mid \cdots \mid \omega_k$, then we define the function P for argument A as $P(A) = \{A, \omega_1, \omega_2, \cdots, \omega_k\}$, while for each terminal a in Σ we have $P(a) = \{a\}$. Note that for each symbol α in V, the value of $P(\alpha)$ is a finite language over the alphabet V that contains α. The containment of α in this value allows us to interpret P as a nested finite substitution; a concept introduced in [10] and to be recalled in §2.

Let us return to errors and their description. Wrongly applying a rule $A \to \omega$, will mean in this paper that an occurrence of A is replaced by an incorrect string ω' instead of the correct string ω. This can be modeled by changing the set $P(A)$ into a fuzzy subset of V^*, and adding a finite number of strings ω' to $P(A)$ with $\mu(\omega'; P(A)) < 1$ for each ω'. This process results in the notion of fuzzy context-free grammar $G = (V, \Sigma, P, S)$ where for each A in N, the set $P(A)$ is now a finite fuzzy subset of V^* rather than an ordinary, or so-called crisp subset. Fuzzy context-free grammars have been introduced in a slightly different, but equivalent way in [14]. So using fuzzy context-free grammars, now we are able to model the situation in which the use of a single correct rule can be replaced by the application of any out of a finite number of incorrect rules.

However, in general there is an infinite number of ways to perform a certain task in an erroneous way and performing a grammatical derivation step is no exception to this rule. But simply replacing the finite fuzzy sets $P(\alpha)$ (for each α in V) by infinite ones will not work, since in that case the language $L(G)$ generated by the resulting grammar G might not even be recursively enumerable [9]. Thus we have to restrain the languages $P(A)$ in some, preferably natural way. The method we use here, originates from [16]; viz. we assume that a family K of fuzzy languages is given in advance, from which we are allowed to take whatever languages we think to be appropriate. Then replacing the finite languages $P(A)$ over V by members from the family K, yields the concept of fuzzy context-free K-grammar. The family K plays the role of parameter in our discussion, and when we take K equal to the constant value FIN_f, the family of finite fuzzy languages, we reobtain the ordinary fuzzy context-free grammars.

Remember that fuzzy sets, fuzzy logic and fuzzy grammars have been applied frequently in linguistics and natural language processing. From the many references we only mention the papers in [18] by the inventor of "fuzziness". The present paper is more a sequel to [14] than to any of the more linguistically oriented papers in [18].

The remaining part of this paper is organized as follows. §2 contains some elementary definitions related to fuzzy languages and in §3 we define fuzzy context-free K-grammars and the fuzzy languages they generate. Properties of these grammars and languages are discussed in §4. Then §5 is devoted to recognizing algorithms for fuzzy context-free languages: we give appropriate modifications of Cock-Younger-Kasami's algorithm and of some recursive descent algorithms. Finally, §6 contains a comparison with an alternative way of describing grammatical errors using fuzzy grammars too [15, 13], and a straightforward generalization of our results from previous sections.

In the next sections emphasis is on the main ideas and on concrete examples; for detailed formal proofs we refer to [6, 7, 8].

2. Preliminaries on Fuzzy Languages

We assume the reader to be familiar with the rudiments of formal language and parsing theory. So for the definitions of context-free grammar, Chomsky Normal Form, and Greibach Normal Form we refer to standard texts like [1, 11, 12].

As mentioned in §1 a *fuzzy language* L over an alphabet Σ is a fuzzy subset of Σ^*, i.e. L is defined by a degree of membership function $\mu_L : \Sigma^* \to [0, 1]$. Actually the function μ_L is the primary notion and L a derived concept, since $L = \{x \in \Sigma^* \mid \mu_L(x) > 0\}$. Henceforth, we write $\mu(x;L)$ rather than $\mu_L(x)$. The *crisp* part of a fuzzy language L is the set $c(L) = \{x \in \Sigma^* \mid \mu(x;L) = 1\}$. A *crisp* (or ordinary) language L is a fuzzy language that satisfies $c(L) = L$.

Next we need a few operations on fuzzy languages: viz. union, intersection, concatenation, and applying a fuzzy function on a fuzzy language. For union and intersection of fuzzy languages L_i ($L_i \subseteq \Sigma_i^*$, $i = 1, 2$) we have

$$\mu(x;L_1 \cup L_2) = \max\{\mu(x;L_1), \mu(x;L_2)\}, \text{ and}$$
$$\mu(x;L_1 \cap L_2) = \min\{\mu(x;L_1), \mu(x;L_2)\}, \tag{1}$$

for all x in $(\Sigma_1 \cup \Sigma_2)^*$, respectively. The *concatenation* [14] of fuzzy languages L_1 and L_2, denoted by $L_1 L_2$, is the fuzzy language satisfying: for all x in $(\Sigma_1 \cup \Sigma_2)^*$,

$$\mu(x;L_1 L_2) = \max\{\min\{\mu(y;L_1), \mu(z;L_2)\} \mid x = yz\}. \tag{2}$$

A *fuzzy relation* R between (ordinary) sets X and Y is a fuzzy subset of $X \times Y$. For fuzzy relations $R \subseteq X \times Y$ and $S \subseteq Y \times Z$, their composition $R \circ S$ is defined by

$$\mu((x,z);R \circ S) = \max\{\min\{\mu((x,y);R), \mu((y,z);S)\} \mid y \in Y\}. \tag{3}$$

A *fuzzy function* $f : X \to Y$ is a fuzzy relation $f \subseteq X \times Y$, satisfying for all x in X: if $\mu((x,y);f) > 0$ and $\mu((x,z);f) > 0$, then $y = z$ and, consequently, we have $\mu((x,y);f) = \mu((x,z);f)$. Note that (3) applies to fuzzy functions as well. But the composition of two functions $f : X \to Y$ and $g : Y \to Z$ is usually written as $g \circ f : X \to Z$ rather than $f \circ g$.

In the sequel we need a function of type $f : X \to \mathcal{P}(X)$ — where $\mathcal{P}(X)$ denotes the power set of the set X — that will be extended to the function $f : \mathcal{P}(X) \to \mathcal{P}(X)$ by $f(S) = \bigcup\{f(x) \mid x \in S\}$ and for each subset S of X,

$$\mu(y;f(S)) = \max\{\min\{\mu(x;S), \mu((x,y);f)\} \mid x \in X\}; \tag{4}$$

cf. e.g. Definition 2.2 below where $X = V^*$ for some alphabet V, and S is a language over V. Fuzzy functions like $f \circ f$, $f \circ f \circ f$, and so on, are now meaningful by (3) and (4).

Next we turn to the notion of family of fuzzy languages.

Definition 2.1. Let Σ_ω be a countably infinite set of symbols. A *family of fuzzy languages* is a set of pairs (L, Σ_L) where L is a fuzzy subset of Σ_L^* and Σ_L is a finite subset of Σ_ω. We assume that the alphabet Σ_L is minimal with respect to L, i.e., a symbol a is a member of Σ_L if and only if a occurs in a word x with $\mu(x;L) > 0$. A family of fuzzy languages K is called *nontrivial* if K contains a language L such that $\mu(x;L) > 0$ for some $x \in \Sigma_\omega^+$. A family is called *crisp* if all its members are crisp languages.

Frequently, we write L instead of (L, Σ_L) for members of a family of (fuzzy) languages, especially when Σ_L is clear from the context. □

Examples of simple, nontrivial families of fuzzy languages, which we will use, are the family FIN_f of finite fuzzy languages, the family ALPHA_f of finite fuzzy languages of which the members have unit length (i.e., these languages are alphabets), and the family SYMBOL_f of singleton languages of unit length. Formally,

$$\text{FIN}_f = \{\{w_1, w_2, \cdots, w_n\} \mid w_i \in \Sigma_\omega^*,\ 1 \le i \le n,\ n \ge 0\},$$
$$\text{ALPHA}_f = \{\Sigma \mid \Sigma \subset \Sigma_\omega, \Sigma \text{ is finite}\}, \text{ and}$$
$$\text{SYMBOL}_f = \{\{\alpha\} \mid \alpha \in \Sigma_\omega\}.$$

The corresponding families of crisp languages are denoted by FIN, ALPHA, and SYMBOL, respectively. Note that the family FIN_f is closed under the operations union, intersection and concatenation, the family ALPHA_f is closed under union and intersection but not under concatenation, whereas SYMBOL_f is not closed under any of these three operations. A similar statement holds for the corresponding crisp families.

Finally, we will consider a more complicated operation on languages; it slightly generalizes a concept from [10].

Definition 2.2. Let K be a family of fuzzy languages. A *nested fuzzy K-substitution* over an alphabet V is a mapping $P : V \to K$ satisfying:

(i) for each α in V, $P(\alpha)$ is a fuzzy K-language over V, and

(ii) P is nested, i.e., $\mu(\alpha;P(\alpha)) = 1$ for each α in V.

The mapping P is extended to words over V by $P(\lambda) = \{\lambda\}$ with $\mu(\lambda;P(\lambda)) = 1$ — where λ denotes the empty word — and $P(\alpha_1\alpha_2 \cdots \alpha_n) = P(\alpha_1)P(\alpha_2)\cdots P(\alpha_n)$ with $\alpha_i \in V$ for all i ($1 \le i \le n$, $n \ge 0$). Finally, P is extended to languages over V by $P(L) = \bigcup\{P(x) \mid x \in L\}$ and, according to (4), for each language L over V,

$$\mu(y;P(L)) = \max\{\min\{\mu(x;L), \mu((x,y);P)\} \mid x \in L\}. \tag{5}$$

A nested fuzzy K-substitution P over V can be iterated, giving rise to a *iterated nested fuzzy K-substitution* over V, i.e., a mapping P^* from languages over V to languages over V, defined by $P^*(L) = \bigcup\{P^n(L) \mid n \ge 0\}$ with $P^{i+1}(L) = P(P^i(L))$ for each $i \ge 0$, and $P^0(L) = L$.

A family K of fuzzy languages is closed under [iterated] nested fuzzy substitution if for each fuzzy language L over some alphabet V, and for each [iterated] nested fuzzy K-substitution over V, we have $P(L) \in K$ [$P^*(L) \in K$, respectively]. □

Note that in Definition 2.2 we used the operations union, concatenation, function application and function composition; cf. (3), (4) and (5) in particular.

Example 2.3. The families SYMBOL_f and ALPHA_f are closed under (iterated) nested fuzzy substitution. On the other hand, although the family FIN_f is closed under nested fuzzy substitution, it is not closed under iterated nested fuzzy substitution. Viz. consider P over $V = \{a,b\}$ with $\mu(a;P(a)) = 1$, $\mu(b;P(b)) = 1$, and $\mu(aba;P(b)) = 0.4$, whereas for all other arguments μ takes the value 0. Now for each fuzzy language L over V that contains at least one word in which a symbol b occurs, the fuzzy language $P^*(L)$ is infinite. Let L be $\{a^n ba^n \mid 0 \le n \le 3\}$ with $\mu(a^n ba^n;L) = 1$ for $n = 0,1$ and $\mu(a^n ba^n;L) = 0.2$ for $n = 2,3$. Then $P^*(L) = \{a^n ba^n \mid n \ge 0\}$ where $\mu(a^n ba^n;P^*(L))$ equals 1 for $n = 0,1$ and 0.4 for all $n \ge 2$. □

3. Fuzzy Context-Free K-grammars

In this section we first discuss grammatical errors by means of a few examples of (fuzzy) context-free grammars. Then we formally define fuzzy context-free K-grammars and the languages they generate.

Example 3.1. Consider the (ordinary) context-free grammar $G = (V,\Sigma,P,S)$ with $V = \Sigma \cup \{S\}$, $\Sigma = \{[,],\langle,\rangle\}$ and P consists of the rules $S \to [S]S \mid \langle S\rangle S \mid \lambda$, where λ denotes the empty word. The language $L(G)$ is called the *Dyck language over two types of parentheses* and it consists of all well-matched sequences over Σ. So $[[\,]\langle\rangle]$ and $\langle\langle\rangle[\,]\langle\rangle\rangle$ are in $L(G)$, but $[\langle\rangle[\,]$ and $[\langle]\rangle$ are not. $L(G)$ plays an important role in the theory of context-free languages, since any context-free language L_0 can be obtained from $L(G)$ by the application of an appropriate non-deterministic finite-state transducer T, i.e. $L_0 = T(L(G))$, where T depends on L_0. As a nested FIN-substitution P looks like $P(S) = \{S, [S]S, \langle S\rangle S, \lambda\}$ and $P(\sigma) = \{\sigma\}$ for each σ in Σ. □

Example 3.2. Let $G_0 = (V, \Sigma, P_0, S)$ be the fuzzy context-free grammar that is equal to G of Example 3.1 except that $P_0(S) = P(S) \cup \{[S\rangle S, [SS\}$, $P_0(\sigma) = \{\sigma\}$ for each σ in Σ, $\mu([S\rangle S; P_0(S)) = 0.9$, $\mu([SS; P_0(S)) = 0.1$, and μ equals 1.0 in all other cases. The string $[S\rangle S$ gives rise to e.g. $\mu([\langle\rangle[\rangle]; L(G_0)) = 0.9$ which is a "tiny mistake" from which it is easy to recover. However, the string $[SS$ causes much more serious problems: we have $\mu([[][]; L(G_0)) = 0.1$, but what is the corresponding correct string? There are three possibilities: $[][][]$, $[[]][]$ and $[[][]]$. So $[[][]$ is considered to be a "capital blunder", when we choose, for instance, δ and Δ equal to 0.2 (cf. §1). □

The next step is that we will allow for an infinite number of ways to make grammatical errors, for which we need grammars with an infinite number of rules.

Example 3.3. Consider the fuzzy context-free grammar $G_1 = (V, \Sigma, P_1, S)$ that is equal to G_0 of Example 3.2 except that $P_1(S) = P(S) \cup \{[^n S\rangle^n S \mid n \geq 1\} \cup \{[SS\}$ with $\mu([^n S\rangle^n S; P_1(S)) = 0.9$ for all $n \geq 1$. It is straightforward to show that $L(G_1) = L(G_0)$, i.e. $\mu(x; L(G_0)) = \mu(x : L(G_1))$ for all x in Σ^*. □

Crisp grammars with an infinite number of rules have been considered previously; e.g. grammars in extended BNF and the grammatical devices in [16, 2, 3]. In the next definition we generalize the fuzzy context-free grammars from [14].

Definition 3.4. Let K be a family of fuzzy languages. A *fuzzy context-free K-grammar* $G = (V, \Sigma, P, S)$ consists of

- an alphabet V (the *alphabet* of G);
- a subset Σ of V (the *terminal alphabet* of G);
- a special nonterminal symbol S (the *initial* or *start symbol* of G);
- a nested fuzzy K-substitution P over V, i.e., a mapping $P : V \to K$ satisfying: for each symbol α in V, $P(\alpha)$ is a fuzzy language over the alphabet V from the family K with $\mu(\alpha; P(\alpha)) = 1$.

The fuzzy *language* generated by G is the fuzzy set $L(G)$ defined by $L(G) = P^*(S) \cap \Sigma^*$. The family of fuzzy languages generated by fuzzy context-free K-grammars is denoted by $A_f(K)$. The corresponding family of crisp languages is denoted by $c(A_f(K))$, i.e., $c(A_f(K)) = \{c(L) \mid L \in A_f(K)\}$. □

Comparing Definition 3.4 with the concept of fuzzy context-free grammar from [14] yields the following differences:

(1) Following 3.4 it is allowed to rewrite terminal symbols.

(2) In 3.4 $L(G)$ is defined in terms of the operations union, intersection, concatenation and (iterated) function application rather than in terms of derivations.

(3) In 3.4 there is an infinite number of rules in P.

Now (1) happens to be a minor point (cf. §4), (2) is just a reformulation, but (3) is the main point. With respect to (2), the language $L(G)$ can also be defined using derivations; cf. [14]. A string x over Σ belongs to $L(G)$ if and only if there exist strings $\omega_0, \omega_1, \cdots, \omega_n$ over V such that $S = \omega_0 \Rightarrow \omega_1 \Rightarrow \omega_2 \cdots \Rightarrow \omega_n = x$. If $A_i \to \psi_i$ $(0 \leq i < n)$ are the respective rules used in this derivation, then

$$\mu(x; L(G)) = \max\{\min\{\mu(\psi_i; P(A_i)) \mid 0 \leq i < n\} \mid S = \omega_0 \Rightarrow^* \omega_n = x\}, \qquad (6)$$

i.e., the maximum is taken over all possible derivations of x from S. If such a derivation is viewed as a chain link of rule applications, its total "strength" equals the strength of its weakest link; hence the min-operation. And $\mu(x; L(G))$ is the strength of the strongest derivation chain from S to x; cf. [14].

Example 3.5. According to the grammar G_1 of Example 3.3 we have the derivation

$$S \Rightarrow [S]S \Rightarrow [[[S\rangle\rangle S]S \Rightarrow [[[\rangle\rangle S]S \Rightarrow [[[\rangle\rangle]S \Rightarrow [[[\rangle\rangle]$$

and $\mu([[[\rangle\rangle]; L(G_1)) = 0.9$ since in the second step we used $\mu([^2 S\rangle^2 S; P_1(S)) = 0.9$ while in all other steps μ has the value 1.0. □

Example 3.6. $A_f(\mathrm{FIN}_f) = \mathrm{CF}_f$, i.e., the family of fuzzy context-free languages from [14]. Hence, $c(A_f(\mathrm{FIN}_f)) = \mathrm{CF}$, the family of (ordinary or crisp) context-free languages. Note that $A_f(\mathrm{SYMBOL}_f) = c(A_f(\mathrm{SYMBOL})) = \{\varnothing\}$, since $S \Rightarrow S$ is the only possible derivation for the corresponding grammars. But $A_f(\mathrm{ALPHA}_f) = \mathrm{ALPHA}_f$, and similarly $c(A_f(\mathrm{ALPHA}_f)) = A_f(\mathrm{ALPHA}) = \mathrm{ALPHA}$ as only sentential forms of length 1 occur in each derivation of the corresponding grammars. □

4. Properties of Fuzzy Context-Free *K*-languages

Throughout this section we restrict our attention to those families K satisfying some minor conditions, collected in

Assumption 4.1. Henceforth, K is a family of fuzzy languages that satisfies:

(1) K contains all crisp SYMBOL-languages ($K \supseteq \mathrm{SYMBOL}$);

(2) K is closed under union with SYMBOL-languages, i.e. for each L from K and each crisp $\{\alpha\}$ from SYMBOL, the fuzzy language $L \cup \{\alpha\}$ also belongs to K;

(3) K is closed under isomorphism ("renaming of symbols"), i.e. for each L over Σ from K and for each bijective mapping $i : \Sigma \to \Sigma_1$ — extended to words and to languages in the usual way — we have that the language $i(L)$ belongs to K. □

Note that the family ALPHA_f is the smallest family satisfying these properties.

Our first result deals with the generating power of fuzzy context-free K-grammars.

Theorem 4.2. Under assumption 4.1, we have $A_f(A_f(K)) = A_f(K)$.

Proof (sketch): The inclusion $A_f(K) \subseteq A_f(A_f(K))$ is easy to establish. Viz. for each L_0 in $A_f(K)$ with fuzzy context-free K-grammar $G = (V, \Sigma, P, S)$ and $L(G) = L_0$, there is a fuzzy context-free $A_f(K)$-grammar $G_0 = (V_0, \Sigma, P_0, S_0)$ with $V_0 = \Sigma \cup \{S_0\}$, $P_0(S_0) = \{S_0\} \cup L(G)$, and $P_0(\sigma) = \{\sigma\}$ for all σ in Σ. Then for each x in Σ^*, we have $\mu(x; L(G_0)) = \mu(x; L(G)) = \mu(x; L_0)$.

To show the converse inclusion, let $G = (V, \Sigma, P, S)$ be a fuzzy context-free $A_f(K)$-grammar. So P is a nested fuzzy $A_f(K)$-substitution over the alphabet V. For each α in V, let $G_\alpha = (V_\alpha, V, P_\alpha, S_\alpha)$ be a fuzzy context-free K-grammar — i.e. each P_α is a nested fuzzy K-substitution over V_α — such that $L(G_\alpha) = P(\alpha)$. We assume that all nonterminal alphabets $V_\alpha - V$ are mutually disjoint. The proof that $L(G) \in A_f(K)$ consists of the following three parts:

(a) Using 4.1.(3) we modify each grammar G_α ($\alpha \in V$) in such a way that $P_\alpha(\beta) = \{\beta\}$ holds for each terminal symbol β in V.

(b) For each nested fuzzy K-substitution P_α over V_α, we define a corresponding nested fuzzy K-substitution Q_α by

$$
\begin{aligned}
Q_\alpha(\beta) &= P_\alpha(\beta) && \text{iff} \quad \beta \in V_\alpha - V \\
Q_\alpha(\beta) &= \{\beta, S_\beta\} && \text{iff} \quad \beta \in V \\
Q_\alpha(\beta) &= \{\beta\} && \text{iff} \quad \beta \in V_1 - V_\alpha
\end{aligned}
$$

with $V_1 = \bigcup\{V_\alpha \mid \alpha \in V\}$. Now we have that $L(G) = \{Q_\alpha \mid \alpha \in V\}^* \cap \Sigma^*$.

(c) Finally, using 4.1.(1)-(3) we reduce the finite set $\{Q_\alpha \mid \alpha \in V\}$ of nested fuzzy K-substitutions over V_1 to an equivalent, single nested fuzzy K-substitution P_0 over an extension V_0 of V_1.

Then for $G_0 = (V_0, \Sigma, P_0, S_S)$, we have $\mu(x; L(G_0)) = \mu(x; L(G))$. Thus, $L(G_0) = L(G)$, and $L(G) \in A_f(K)$, i.e., $A_f(A_f(K)) \subseteq A_f(K)$. □

For a complete constructive proof of 4.2 we refer to [6, 7]. Though from a mathematical point of view 4.2 is quite appealing, we turn to the special case $K = \mathrm{FIN}_f$ in order to resume our discussion on errors and fuzzy context-free grammars.

Corollary 4.3. $A_f(A_f(\mathrm{FIN}_f)) = A_f(\mathrm{CF}_f) = A_f(\mathrm{FIN}_f) = \mathrm{CF}_f$. □

Corollary 4.3 provides us the limit of deriving grammatical errors within the framework of fuzzy context-free grammars. Viz. we may extend the sets $P(\alpha)$ (for each α in the alphabet V) to infinite sets, as long as the resulting sets $P(\alpha)$ still constitute fuzzy context-free languages over V. Only in this way we are able to model the case of an infinite number of possible grammatical errors. Of course, during each derivation only a finite choice out of this infinity of possible errors will be made.

Though the construction in (second part of) the proof of 4.2 is applicable to each fuzzy context-free $A_f(K)$, sometimes an ad hoc construction may result in a simpler grammar.

Example 4.4. Consider the fuzzy context-free $A_f(\mathrm{FIN}_f)$-grammar or fuzzy context-free CF_f-grammar G_1 of Example 3.3. We will construct an equivalent fuzzy context-free FIN_f-grammar $G_2 = (V_2, \Sigma, P_2, S)$. Let $V_2 = \{A\} \cup V_1$, $P_2(S) = P(S) \cup \{[SS, AS\}$, $P_2(A) = \{A, [A\rangle, [S\rangle\}$, $\mu(AS; P_2(S)) = \mu([A\rangle; P_2(A)) = \mu([S\rangle; P_2(A)) = 0.9$ and everything else is as in Example 3.3. Then $\mu(x; L(G_2)) = \mu(x; L(G_1))$ for all x in Σ^*, i.e., $L(G_2) = L(G_1)$. □

We conclude this section with some mathematical consequences of 4.2 and 4.3 for which we need the following fuzzy analogue to the notion of full super-AFL [10].

Definition 4.5. A nontrivial family K of fuzzy languages is called a *full super-AFFL* (i.e., *full super-Abstract Family of Fuzzy Languages*) if K is closed under

- finite fuzzy substitution (i.e. FIN_f-substitution);
- intersection with fuzzy regular languages;
- iterated nested fuzzy substitution. □

From closure under these three operations, closure under many other operations well known in formal language theory follows: closure under union, concatenation, Kleene *, homomorphism, inverse homomorphism, substitution, nondeterministic finite-state transductions, and so on; cf. [10, 2, 4] and also [17].

Theorem 4.6. [7]

(1) Let K be a nontrivial family of fuzzy languages closed under finite fuzzy substitution and under intersection with fuzzy regular languages. Then $A_f(K)$ is a full super-AFFL, and, in particular, it is the smallest full super-AFFL that includes the family K.

(2) Each full super-AFFL includes the family CF_f of fuzzy context-free languages.

(3) The family CF_f is the smallest full super-AFFL. □

The proof of 4.6 heavily relies on Theorem 4.2 and Corollary 4.3; cf. [7]. Comparing 4.6 with results in [10, 2, 4] yields that the family of fuzzy context-free languages possesses closure properties very similar to those of the (ordinary or crisp) context-free languages.

5. Recognizing Fuzzy Context-Free Languages

In this section we give a few algorithms for recognizing fuzzy context-free languages. When a fuzzy context-free language has been specified by a fuzzy context-free CF_f-grammar (Corollary 4.3), we must first transform that grammar into an equivalent fuzzy context-free FIN_f-grammar by means of the construction in the proof of Theorem 4.2. Next we must transform the resulting grammar into Chomsky or Greibach Normal Form, using results from [14], before we can apply the algorithms from this section.

The first algorithm is a modification of Cocke–Younger–Kasami's algorithm (or CYK-algorithm for short); cf. Algorithm 5.2 below. In e.g. [1, 11, 12] the CYK-algorithm is given in terms of nested **for**-loops that fill an upper-triangular matrix. Here we start from an alternative functional version from [5] which has some interesting features: it omits implementation details like the data structure, reference to the indices of matrix entries and to the length of the input string; cf., e.g., Algorithm 12.4.1 in [11] and Algorithm 5.1 below.

Algorithm 5.1. Let $G = (V, \Sigma, P, S)$ be a λ-free context-free grammar in Chomsky Normal Form and let w be in Σ^+. Define functions $f : \Sigma^+ \to \mathcal{P}(N^+)$ and $g : \mathcal{P}(N^+) \to \mathcal{P}(N)$ by:

- For each w in Σ^+ the function f is defined as the finite substitution generated by

$$f(a) = \{A \mid a \in P(A)\} \tag{7}$$

and extended to words over Σ by

$$f(w) = f(a_1)f(a_2) \cdots f(a_n) \text{ if } w = a_1 a_2 \cdots a_n \quad (a_k \in \Sigma,\ 1 \le k \le n). \tag{8}$$

- For each A in N we define $g(A) = \{A\}$ and for each ω in N^+ with $|\omega| \ge 2$ we have

$$g(\omega) = \bigcup \{g(\chi) \otimes g(\eta) \mid \chi, \eta \in N^+,\ \omega = \chi\eta\} \tag{9}$$

where for each X and Y in $\mathcal{P}(N)$ the binary operation $\otimes$ is defined by

$$X \otimes Y = \{A \mid BC \in P(A), \text{ with } B \in X \text{ and } C \in Y\}. \tag{10}$$

- For each (finite) language M over N, $g(M)$ is defined by

$$g(M) = \bigcup \{g(\omega) \mid \omega \in M\}. \tag{11}$$

Finally, compute $g(f(w))$ and determine whether S belongs to $g(f(w))$.

Clearly, we have $w \in L(G)$ if and only if $S \in g(f(w))$. □

From this functional version of the CYK-algorithm it is easy to derive an algorithm for recognizing fuzzy context-free languages.

Algorithm 5.2. Let $G = (V, \Sigma, P, S)$ be a λ-free fuzzy context-free grammar in Chomsky Normal Form and let w be in Σ^+. Extend (7)–(11) in Algorithm 5.1 with

$$\mu(A; f(a)) = \mu(a; P(A)), \tag{7'}$$

$$\mu(A; X \otimes Y) = \min\{\mu(BC; P(A)), \mu(B; X), \mu(C; Y)\}, \tag{9'}$$

$$\mu(A; g(\omega)) = \max\{\mu(A; g(\chi) \otimes g(\eta)) \mid \chi, \eta \in N^+,\ \omega = \chi\eta\}, \tag{10'}$$

whereas corresponding equalities for (8) and (11) follow from the definitions of concatenation and finite union, respectively; cf. §2. Finally, compute $\mu(S; g(f(w)))$.

Then, we have $\mu(w; L(G)) = \mu(S; g(f(w)))$. □

Example 5.3. Consider the fuzzy context-free grammar $G_3 = (V_3, \Sigma, P_3, S)$ with $V_3 = \Sigma \cup \{S, A, B, C, D, E, F\}$, P_3 consists of the rules

$$\begin{array}{lll} S \to SS \mid AC \mid BC \mid DF \mid EF \mid AF \mid BF \mid BS \mid [\,, & & \\ A \to BS\,, & B \to [\,, & C \to]\,, \\ D \to ES\,, & E \to \langle\,, & F \to \rangle\,, \end{array}$$

where $\mu(AF; P_3(S)) = \mu(BF; P_3(S)) = 0.9$, $\mu(BS; P_3(S)) = \mu([; P_3(S)) = 0.1$ and μ has value 1.0 in all other cases. The grammar G_3 is λ-free, in Chomsky Normal Form, and equivalent (modulo the empty word λ) to G_0 from Example 3.2.

Applying Algorithm 5.2 with, e.g., input equal to $[\,]\langle\rangle$, yields

$$\mu([\,]\langle\rangle; L(G_3)) = \mu(S; g(f([\,]\langle\rangle))) = \mu(S; g(\{B,S\}CEF)) = \mu(S; g(\{B,S\}) \otimes g(CEF) \cup$$
$$\cup\, g(\{B,S\}C) \otimes g(EF) \cup g(\{B,S\}CE) \otimes g(F)) = \cdots = \mu(S; S) = 1.0$$

where we write, as usual, X for a singleton set $\{X\}$. Similarly, for input equal to $[[\,]\rangle$, we get

$$\mu([[\,]\rangle; L(G_3)) = \mu(S; g(f([[\,]\rangle))) = \mu(S; g(\{B,S\}\{B,S\}CF)) =$$
$$= \mu(S; g(\{B,S\}) \otimes g(\{B,S\}CF) \cup g(\{B,S\}\{B,S\}) \otimes g(CF) \cup$$
$$\cup\, g(\{B,S\}\{B,S\}C) \otimes g(F)) = \cdots = 0.9$$

□

Algorithms 5.1 and 5.2 are bottom-up algorithms for recognizing λ-free (fuzzy) context-free languages in Chomsky Normal Form. Functional versions of top-down ("recursive descent") algorithms for crisp context-free languages have been introduced in [5], from which we recall Definition 5.4 and Algorithm 5.5. In Algorithm 5.6 we give a modification of 5.5 which results in a recursive descent recognizer for fuzzy context-free languages in Chomsky Normal Form.

Definition 5.4. For each context-free grammar $G = (V,\Sigma,P,S)$ with $N = V - \Sigma$, the set $T(\Sigma,N)$ of *terms* over (Σ,N) is the smallest set defined by

(a) λ is a term in $T(\Sigma,N)$ and each a $(a \in \Sigma)$ is a term in $T(\Sigma,N)$.

(b) For each A in N and each term t in $T(\Sigma,N)$, $A(t)$ is a term in $T(\Sigma,N)$.

(c) If t_1 and t_2 are in $T(\Sigma,N)$, then their concatenation t_1t_2 is a term in $T(\Sigma,N)$ too. □

Note that for any two sets of terms S_1 and S_2 $(S_1, S_2 \subseteq T(\Sigma,N))$ the set S_1S_2, defined by $S_1S_2 = \{t_1t_2 \mid t_1 \in S_1,\ t_2 \in S_2\}$, is also a set of terms over (Σ,N).

Algorithm 5.5. Let $G = (V,\Sigma,P,S)$ be a λ-free context-free grammar in Chomsky Normal Form and let w be a string in Σ^+. Each nonterminal symbol A in N is considered as a function from $\Sigma^* \cup \{\perp\}$ to $\mathcal{P}(T(\Sigma,N))$ defined as follows. (The symbol $\perp$ will be used to denote "undefined".) First, $A(\perp) = \varnothing$ and $A(\lambda) = \{\lambda\}$ for each A in N. If the argument x of A is a word of length 1 (i.e. x is in Σ) then

$$A(x) = \{\lambda \mid x \in P(A)\} \qquad (x \in \Sigma) \tag{12}$$

and in case the length $|x|$ of the word x is 2 or more, then

$$A(x) = \bigcup \{B(y)C(z) \mid BC \in P(A),\ y,z \in \Sigma^+,\ x = yz\}. \tag{13}$$

Finally, we compute $S(w)$ and determine whether λ belongs to $S(w)$.

It is straightforward to show that $w \in L(G)$ if and only if $\lambda \in S(w)$. □

Algorithm 5.6. Let $G = (V,\Sigma,P,S)$ be a λ-free fuzzy context-free grammar in Chomsky Normal Form and let w be a string in Σ^+. For all A in N, $\mu(\lambda;A(\lambda)) = 1$ and $\mu(t;A(\perp)) = 0$ for each t in $\mathcal{P}(T(\Sigma,N))$. Extend (12)–(13) in Algorithm 5.5 with

$$\mu(\lambda;A(x)) = \mu(x;P(A)) \qquad (x \in \Sigma), \tag{12'}$$

$$\mu(\lambda;A(x)) = \max\{\min\{\mu(BC;P(A)), \mu(\lambda;B(y)), \mu(\lambda;C(z))\} \mid BC \in P(A),\ y,z \in \Sigma^+,\ x = yz\}. \tag{13'}$$

Finally, we compute $\mu(\lambda;S(w))$. Then we have $\mu(w;L(G)) = \mu(\lambda;S(w))$. □

Example 5.7. Applying Algorithm 5.6 to the fuzzy context-free grammar G_3 of Example 5.3 results in

$$\begin{aligned}\mu(\langle\rangle[\,];L(G_3)) &= \mu(\lambda;S(\langle\rangle[\,])) = \\ &= \mu(\lambda;S(\langle\rangle[)S(]) \cup S(\langle\rangle)S([\,]) \cup S(\langle)S(\rangle[\,]) \cup \\ &\qquad A(\langle\rangle[)C(]) \cup A(\langle\rangle)C([\,]) \cup A(\langle)C(\rangle[\,]) \cup \\ &\qquad B(\langle\rangle[)C(]) \cup B(\langle\rangle)C([\,]) \cup B(\langle)C(\rangle[\,]) \cup \\ &\qquad D(\langle\rangle[)F(]) \cup D(\langle\rangle)F([\,]) \cup D(\langle)F(\rangle[\,])) \cup \\ &\qquad E(\langle\rangle[)F(]) \cup E(\langle\rangle)F([\,]) \cup E(\langle)F(\rangle[\,])) = \cdots = 1\end{aligned}$$

$$\mu([\langle\rangle;L(G_3)) = \mu(\lambda;S([\langle\rangle)) = \cdots = 0.1$$

$$\mu(\langle]\rangle;L(G_3)) = \mu(\lambda;S(\langle]\rangle)) = \cdots = 0$$ □

Finally, we give analogues of Algorithms 5.5 and 5.6 based on Greibach 2-form (viz. Algorithms 5.8 and 5.9) which are slightly more efficient then 5.5 and 5.6, respectively. Recall that a λ-free context-free grammar is in *Greibach 2-form* if its rules possess one of the forms: $A \to aBC$, $A \to aB$ and $A \to a$ $(a \in \Sigma,\ A,B,C \in N)$.

Algorithm 5.8. Let $G = (V,\Sigma,P,S)$ be a λ-free context-free grammar in Greibach 2-form and let w be a string in Σ^+. The algorithm is as Algorithm 5.5 except that (13) is

replaced by

$$A(x) = \bigcup \{B(y)C(z) \mid aBC \in P(A),\ y,z \in \Sigma^+,\ x = ayz\} \cup \tag{14}$$
$$\bigcup \{B(y) \mid aB \in P(A),\ y \in \Sigma^+,\ x = ay\}.$$

Still we have that $w \in L(G)$ if and only if $\lambda \in S(w)$. □

Algorithm 5.9. Let $G = (V,\Sigma,P,S)$ be a λ-free fuzzy context-free grammar in Greibach 2-form and let w be a string in Σ^+. For all A in N, $\mu(\lambda;A(\lambda)) = 1$ and $\mu(t;A(\perp)) = 0$ for each t in $\mathcal{P}(T(\Sigma,N))$. Extend (14) in Algorithm 7.5 with

$$\mu(\lambda;A(x)) = \mu(\lambda;A'(x) \cup A''(x)) \qquad \text{with} \tag{14'}$$
$$\mu(\lambda;A'(x)) = \max\{\min\{\mu(aBC;P(A)),\ \mu(\lambda;B(y)),\ \mu(\lambda;C(z))\} \mid aBC \in P(A),\ y,z \in \Sigma^+,\ x = ayz\}, \tag{14'}$$
$$\mu(\lambda;A''(x)) = \max\{\min\{\mu(aB;P(A)),\ \mu(\lambda;B(y))\} \mid aB \in P(A),\ y \in \Sigma^+,\ x = ay\}. \tag{14'}$$

Finally, compute $\mu(\lambda;S(w))$. Then $\mu(w;L(G)) = \mu(\lambda;S(w))$. □

Example 5.10. Let $G_4 = (V_4,\Sigma,P_4,S)$ be the fuzzy context-free grammar with $V = \Sigma \cup \{S,C,F\}$, and P_4 consists of rules which are displayed with their degree of membership in the following table.

set	elements	degree
$P_4(S)$	$S, [SCS, \langle SFS, [CS, \langle FS, [SC, \langle SF, [C, \langle F$	1.0
	$[SFS, [FS, [SF, [F$	0.9
	$[SS, [S, [$	0.1
$P_4(C)$	$C,]$	1.0
$P_4(F)$	$F, \rangle$	1.0

Applying Algorithm 5.9 yields $\mu(\lambda;C(])) = \mu(\lambda;F(\rangle)) = 1.0$, $\mu(\lambda;S([)) = 0.1$, and

$$\begin{aligned} S(w) = {} & \bigcup\{S(x)C(y)S(z) \mid w = [xyz\} \cup \bigcup\{C(x)S(y) \mid w = [xy\} \cup \\ & \bigcup\{S(x)F(y)S(z) \mid w = \langle xyz\} \cup \bigcup\{F(x)S(y) \mid w = \langle xy\} \cup \\ & \bigcup\{S(x)C(y) \mid w = [xy\} \cup \bigcup\{S(x)F(y) \mid w = \langle xy\} \cup \\ & \bigcup\{S(x)F(y)S(z) \mid w = [xyz\} \cup \bigcup\{F(x)S(y) \mid w = [xy\} \cup \\ & \bigcup\{S(x)F(y) \mid w = [xy\} \cup \bigcup\{S(x)S(y) \mid w = [xy\} \cup \\ & \cup\, C([\setminus w) \cup F(\langle \setminus w) \cup F([\setminus w) \cup S([\setminus w) \end{aligned}$$

where x,y and z are nonempty strings over Σ, and $u \setminus v = w$ if $v = uw$, and $\perp$ otherwise ($u,v,w \in \Sigma^*$). Then we have

$$\begin{aligned} \mu(\lambda;S([\langle\rangle])) &= \mu(\lambda; \cdots \cup S(\langle\rangle)C(]) \cup \cdots) = \\ &= \mu(\lambda; \cdots \cup S(\langle\rangle) \cup \cdots) = \mu(\lambda; \cdots \cup F(\rangle) \cup \cdots) = 1.0 \end{aligned}$$

where numerous non-productive terms have been omitted. Similarly,

$$\mu(\lambda;S([[]])) = \mu(\lambda; \cdots \cup S([)C(]) \cup S([\setminus [[]) \cup \cdots) = 0.1$$ □

Of course, the Greibach 2-form is by no means essential; the transformation to this normal form gives rise to numerous additional rules and less transparent algorithms. For instance, a λ-free version of G_0 from Example 3.2 will result in an algorithm that is simpler than 5.10.

6. Concluding Remarks

We showed that using fuzzy context-free K-grammars we are able to model the case in which at each derivation step a choice from an infinity of possible grammatical errors is made. From Theorem 4.2 and Corollary 4.3 it followed that in order to stay within the framework of fuzzy context-free language generation this choice should be limited to a fuzzy context-free language. However, to apply the recognition algorithms from §5, these fuzzy context-free CF_f-grammars should be transformed into equivalent fuzzy context-free FIN_f-grammars. These recognition algorithms, which are straightforward modifications of existing ones, are robust in a very primitive sense; viz. since they compute the membership function, they can distinguish between "tiny mistakes" and "capital blunders".

Our approach in describing grammatical errors has a global character: a right-hand side ω of a grammar rule may be replaced erroneously by a completely different string ω'. In [15] an alternative way of describing errors — using fuzzy context-free grammars too — is given. Here ω' is restricted to those strings that are obtainable by simple edit operations (deletion, insertion, and substitution of symbols) from ω, and these operations are performed on nonterminal symbols only. In a companion paper [13] this approach is extended to context-sensitive grammars as well, but both papers are restricted to the discussion of simple examples rather than proving general results.

The definition of fuzzy context-free K-grammar can be slightly generalized: viz. instead of a single nested fuzzy K-substitutions we may allow a finite number of such substitutions [7]. Under assumption 4.1 it is possible to reduce this finite number to an equivalent single nested fuzzy K-substitution; cf. the last part of the proof of 4.2.

Of much more practical interest is another modification / generalization. From a certain point of view the model discussed in this paper is rather trivial: to each grammar rule we associate a real number in between 0 and 1, and these numbers are propagated by means of the min-operation (of course, without any alternation) to a string derived by the fuzzy context-free grammar. So making the very same error twice is as bad as making it a single time. Intuitively, one would prefer that the degree of membership in the first case is strictly lower than the one due to the single error. This can be achieved by deviating from the original definition of fuzzy grammar from [14]. When we replace the min-operation in (2)–(5) and, consequently, in (6), (9′), (13′) and (14′) — but not in (1) — by the multiplication operation, we are able to model this accumulation process of errors.

Example 6.1. When we replace the min-operation by multiplication at the appropriate places, Algorithm 5.2 applied to the grammar G_3 of Example 5.3 yields, for instance, $\mu([[\rangle[]\rangle;L(G_3)) = \cdots = 0.81$, $\mu([[\rangle[\rangle\rangle;L(G_3)) = \cdots = 0.729$, $\mu([[\rangle[];L(G_3)) = \cdots = 0.08$ and $\mu([[\rangle[;L(G_3)) = \cdots = 0.008$. □

In this way the occurrence of many "tiny mistakes" may result in the end in something that resembles a "capital blunder".

Acknowledgements. I am indebted to Rieks op den Akker and Anton Nijholt for their critical remarks.

References

1. A.V. Aho & J.D. Ullman: *The Theory of Parsing, Translation and Compiling – Volume I: Parsing* (1972), Prentice-Hall, Englewood Cliffs, NJ.
2. P.R.J. Asveld: *Iterated Context-Independent Rewriting – An Algebraic Approach to Families of Languages*, (1978), Ph.D. Thesis, Dept. of Appl. Math., Twente University of Technology, Enschede, The Netherlands.
3. P.R.J. Asveld: Abstract grammars based on transductions, *Theoret. Comput. Sci.* **81** (1991) 269–288.
4. P.R.J. Asveld: An algebraic approach to incomparable families of formal languages, pp. 455-475 in G. Rozenberg & A. Salomaa (eds.): *Lindermayer Systems – Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology* (1992), Springer-Verlag, Berlin, etc.

5. P.R.J. Asveld: An alternative formulation of Cocke–Younger–Kasami's algorithm, *Bull. Europ. Assoc. for Theoret. Comp. Sci.* (1994) No. 53, 213–216.
6. P.R.J. Asveld: Towards robustness in parsing — Fuzzifying context-free language recognition, Memoranda Informatica 95-08, Dept. of Comp. Sci., Twente University of Technology, Enschede, The Netherlands. To appear in *Proc. 2nd Internat. Conf. on Developments in Language Theory* (1995).
7 P.R.J. Asveld: Fuzzy context-free languages — Part I: Generalized fuzzy context-free grammars, Memoranda Informatica 95-??, Dept. of Comp. Sci., Twente University of Technology, Enschede, The Netherlands (In preparation).
8. P.R.J. Asveld: Fuzzy context-free languages — Part II: Recognition Algorithms, Memoranda Informatica 95-??, Dept. of Comp. Sci., Twente University of Technology, Enschede, The Netherlands (In preparation).
9. G. Gerla: Fuzzy grammars and recursively enumerable fuzzy languages, *Inform. Sci.* **60** (1992) 137-143.
10. S.A. Greibach: Full AFL's and nested iterated substitution, *Inform. Contr.* **16** (1970) 7–35.
11. M.A. Harrison: *Introduction to Formal Language Theory* (1978), Addison-Wesley, Reading, Mass.
12. J.E. Hopcroft & J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation* (1979), Addison-Wesley, Reading, Mass.
13. M. Inui, W. Shoaff, L. Fausett & M. Schneider: The recognition of imperfect strings generated by fuzzy context-sensitive grammars, *Fuzzy Sets and Systems* **62** (1994) 21-29.
14. E.T. Lee & L.A. Zadeh: Note on fuzzy languages, *Inform. Sci.* **1** (1969) 421-434.
15. M. Schneider, H. Lim & W. Shoaff: The utilization of fuzzy sets in the recognition of imperfect strings, *Fuzzy Sets and Systems* **49** (1992) 331-337.
16. J. van Leeuwen: A generalization of Parikh's theorem in formal language theory, pp. 17-26 in: J. Loeckx (ed.): *2nd ICALP*, Lect. Notes in Comp. Sci. 14 (1974), Springer-Verlag, Berlin, etc.
17. W. Wechler: *The Concept of Fuzziness in Automata and Language Theory* (1978), Akademie-Verlag, Berlin.
18. R.R. Yager, R.M. Tong, S. Ovchinnikov & H.T. Nguyen (eds.): *Fuzzy Sets and Applications – Selected Papers by L.A. Zadeh* (1987), J. Wiley, New York.