Towards Robustness in Parsing

—— Fuzzifying 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

Abstract – We discuss the concept of robustness with respect to parsing a context-free language. Our approach is based on the notions of fuzzy language, (generalized) fuzzy context-free grammar and parser / recognizer for fuzzy languages. As concrete examples we consider a robust version of Cocke–Younger–Kasami's algorithm and a robust kind of recursive descent recognizer.

Keywords and phrases: fuzzy language, fuzzy context-free grammar, fuzzy context-free K-grammar, parsing / recognition of fuzzy languages, Cocke–Younger–Kasami's algorithm, recursive descent.

1. Introduction

Informally, we call a context-free language parser or recognizer robust if it is able to deal with small errors. But what is a small error? An input for a parsing or recognizing algorithm is either accepted (when it belongs to the language under consideration) or rejected (when it is outside this language). Thus in this traditional approach there is no room for subtleties like a distinction between a "tiny mistake" and a "capital blunder".

Fortunately, the framework of fuzzy language theory enables us to make such a distinction. Here each language L_0 over an alphabet Σ is a fuzzy subset of the set Σ^*, i.e. the degree of membership of a string x over Σ is determined by a function $\phi : \Sigma^* \to [0, 1]$ instead of the usual characteristic function $\phi : \Sigma^* \to \{0, 1\}$. So the set $\{0, 1\}$ with two elements has been changed into the continuous interval $[0, 1]$ and now $\phi_{L_0}(x)$ can take any real value in between 0 and 1. Thus this concept allows for both "tiny mistakes" (i.e., strings x with $1 - \delta \leq \phi_{L_0}(x) < 1$) and "capital blunders" (strings x with $0 \leq \phi_{L_0}(x) < \Delta$) with respect to L_0, once we made an appropriate choice for δ and Δ. In this framework of fuzzy languages we will consider two problems related to robustness in parsing / recognizing a context-free language.

The first question we address is the type of errors we allow in the input of the parser (or recognizer) and the way we produce these errors. In the approach we follow, the choice of a fuzzy context-free grammar (§2) or a generalized fuzzy context-free grammar (§3) is an obvious one. The latter one turns out to be one of the most general ways to describe context-free languages with both correct as well as erroneous sentences generated by a single fuzzy grammar; cf. Corollary 3.4.

The second problem we discuss is the concept of robustness in parsing or recognizing context-free languages (§4). In this paper we restrict ourselves to recognizing rather than parsing, but our main results can be easily extended to corresponding robust parsing algorithms. In §4 we provide a robust version of Cocke–Younger–Kasami's recognition algorithm, whereas §5 is devoted to a robust recursive descent recognizer.

The remaining two sections contain preliminaries on languages, grammars and their fuzzy counterparts (§2), and concluding remarks (§6).

2. Definitions

We assume familiarity with the rudiments of formal languages, grammars and parsing; cf. e.g. [1, 7, 8]. Fuzzy languages and grammars have been introduced in [10].

Let $G = (V, \Sigma, P, S)$ be a context-free grammar with alphabet V, terminal alphabet Σ, set of productions P and start symbol S. The set of nonterminal symbols of G is $N = V - \Sigma$. The empty word is denoted by λ. A context-free grammar is called λ-*free* if the right-hand side of each production is nonempty.

Remember that a λ-free context-free grammar $G = (V, \Sigma, P, S)$ is in *Chomsky Normal Form* if $P \subseteq N \times (\Sigma \cup N \times N)$. Similarly, a λ-free context-free grammar $G = (V, \Sigma, P, S)$ is in *Greibach 2-form* if $P \subseteq \Sigma \times (\{\lambda\} \cup N \cup N \times N)$.

A *fuzzy language* L_0 over an alphabet Σ is a fuzzy subset of Σ^*, i.e. it is a pair (L_0, ϕ_{L_0}) where ϕ_{L_0} is a function $\phi_{L_0} : \Sigma^* \to [0, 1]$, the so-called *degree of membership function* of L_0, and $L_0 = \{ w \in \Sigma^* \mid \phi_{L_0}(w) > 0 \}$. Let L_0 be a fuzzy language over Σ. The *crisp language* $c(L_0)$ induced by L_0 —also called the *crisp part* of L_0— is the subset $\{ w \in \Sigma^* \mid \phi_{L_0}(w) = 1 \}$ of Σ^*. So each ordinary language L_0 coincides with its crisp part $c(L_0)$. Therefore an ordinary language will also be called a *crisp* language. Frequently, we will write $\phi(x; L_0)$ instead of $\phi_{L_0}(x)$ for x in Σ^*.

Remark. Since the function ϕ has as its codomain the interval $[0, 1]$, each real number from this interval may occur as value for some argument x. However, using non-computable reals as value or as a threshold may give rise to undecidable problems; cf. [5] for details. Therefore we restrict ourselves in the sequel to computable (or even to rational) elements of $[0, 1]$ only. □

Next we consider operations on fuzzy languages. The operations union and intersection for fuzzy languages are defined as usual in fuzzy set theory; cf. [10]. Viz. let (L_1, ϕ_{L_1}) and (L_2, ϕ_{L_2}) be fuzzy languages, then for the *union* of the fuzzy languages L_1 and L_2, denoted by $(L_1 \cup L_2, \phi_{L_1 \cup L_2})$ or $L_1 \cup L_2$ for short, we have

$$\phi(x; L_1 \cup L_2) = \max \{ \phi(x; L_1), \phi(x; L_2) \}, \text{ for all } x \text{ in } \Sigma^*.$$

Similarly, for the *intersection* of the fuzzy languages L_1 and L_2, denoted by $(L_1 \cap L_2, \phi_{L_1 \cap L_2})$ or $L_1 \cap L_2$ for short, we have

$$\phi(x; L_1 \cap L_2) = \min \{ \phi(x; L_1), \phi(x; L_2) \}, \text{ for all } x \text{ in } \Sigma^*.$$

Finally, we consider the operation of concatenation as in [10]; for the *concatenation* of fuzzy languages L_1 and L_2, denoted by $(L_1 L_2, \phi_{L_1 L_2})$ or $L_1 L_2$ for short, holds

$$\phi(x; L_1 L_2) = \sup \{ \min \{ \phi(y; L_1), \phi(z; L_2) \} \mid x = yz \}, \text{ for all } x \text{ in } \Sigma^*.$$

Once we have defined this operation it is easy to define the operation of *Kleene* $*$ by $L_1^* = \{\lambda\} \cup L_1 \cup L_1 L_1 \cup L_1 L_1 L_1 \cup \cdots$ where we require that $\phi(\lambda; L_1^*) = 1$.

The notion of fuzzy context-free grammar has been introduced in [10]. In Definition 2.1 we define fuzzy context-free grammars in a different way, but it is easy to show that 2.1 is equivalent to the definition in [10]. To this end let $G = (V, \Sigma, P, S)$ be an ordinary context-free grammar. For each α in V we define

$$P(\alpha) \;=\; \{\, \omega \mid \alpha \to \omega \in P \,\} \cup \{\alpha\},$$

i.e. $P(\alpha)$ is the set consisting of α together with all right-hand sides of those productions in P with left-hand side equal to α. Thus for each α, $P(\alpha)$ is a finite language over V that contains α. And $P(\alpha)$ equals $\{\alpha\}$ whenever α belongs to Σ.

So P may be considered as a mapping from V to finite languages over V; it can be extended to words over V by $P(\lambda) = \{\lambda\}$, $P(\alpha_1 \cdots \alpha_n) = P(\alpha_1) \cdots P(\alpha_n)$ where $\alpha_i \in V$ $(1 \le i \le n)$, and to languages L over V by $P(L) = \bigcup\{P(x) \mid x \in L\}$.

Since $\alpha \in P(\alpha)$ for each α in V, P is called a *nested finite substitution* over V [6, 12, 2, 3]. Such a nested finite substitution can be *iterated*, viz. $P^0(x) = \{x\}$, $P^{i+1}(x) = P(P^i(x))$, and $P^*(x) = \bigcup\{P^i(x) \mid i \ge 0\}$. Then for each context-free grammar $G = (V, \Sigma, P, S)$, we have $L(G) = P^*(S) \cap \Sigma^*$.

Definition 2.1. A *fuzzy context-free grammar* G is a context-free grammar $G = (V, \Sigma, P, S)$ where for each α in V, $P(\alpha)$ is a fuzzy subset of V^* satisfying

(i) $\phi(\alpha; P(\alpha)) = 1$, i.e., P is *nested*,

(ii) the *support* of $P(\alpha)$, i.e. the set $\{\omega \mid \phi(\omega; P(\alpha)) \neq 0\}$, is finite, and

(iii) the support of $P(\alpha)$ equals $\{\alpha\}$ in case α belongs to Σ.

The (fuzzy context-free) *language* generated by G is the fuzzy set $L(G)$ defined by $L(G) = P^*(S) \cap \Sigma^*$. □

In this latter expression all operations involved are operations on fuzzy sets (intersection, and both union and concatenation via P^*), although Σ^* is a crisp set.

Note that, if we replace in a fuzzy context-free grammar each fuzzy set $P(\alpha)$ by a crisp language over V, then we obtain an ordinary context-free grammar.

The language generated by a fuzzy context-free grammar G can also be defined in terms of derivations consisting of production rules that are applied consecutively; cf. [10]. A string x over Σ belongs to the language $L(G)$ if and only if there exists 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 $A_i \to \psi_i$ $(0 \le i < n)$ are the respective productions used in this derivation, then the degree of membership of x in $L(G)$ is

$$\phi(x; L(G)) \;=\; \sup\{\, \min\{\phi(\psi_i; P(A_i)) \mid 0 \le i < n\} \mid S = \omega_0 \Rightarrow^* \omega_n = x \,\},$$

i.e., the supremum is taken over all possible derivations of x from S. If such a derivation is viewed as a chain link of production applications, its total "strength" equals the strength of its weakest link; hence the min-operation. And $\phi(x; L(G))$ is the strength of the strongest derivation chain from S to x; cf. [10].

In the sequel $|w|$ denotes the length of the string w.

Example 2.2. Consider the fuzzy context-free grammar $G_0 = (V, \Sigma, P_0, S)$ with $N = V - \Sigma = \{S, A, B\}$, $\Sigma = \{a, b\}$, and P_0 is defined by

$P_0(S) = \{S, AB, BA, AA, BB\}$,

$P_0(A) = \{A, AS, SA, a\}$,

$P_0(B) = \{B, BS, SB, b\}$,
$P_0(\sigma) = \{\sigma\}$ if $\sigma \in \Sigma$.

The degrees of membership are $\phi(AA; P_0(S)) = 0.1$, $\phi(BB; P_0(S)) = 0.9$, and equal to 1 in all other instances. The crisp language $c(L(G))$ is generated by the (ordinary) context-free grammar $G_1 = (V, \Sigma, P_1, S)$ where P_1 is defined by

$P_1(S) = \{S, AB, BA\}$,
$P_1(A) = \{A, AS, SA, a\}$,
$P_1(B) = \{B, BS, SB, b\}$,
$P_1(\sigma) = \{\sigma\}$ if $\sigma \in \Sigma$.

It is straightforward to show that

- $c(L(G_0)) = L(G_1) = \{w \mid w \in \{a, b\}^+, \ \#_a(w) = \#_b(w)\}$, where $\#_\sigma(w)$ denotes the number of times that the symbol σ occurs in the string w,
- $\phi(w; L(G_0)) = 0.1$ if and only if $\#_a(w) \geq \#_b(w) + 2$ and $|w|$ is even ($w \in \{a, b\}^+$),
- $\phi(w; L(G_0)) = 0.9$ if and only if $\#_b(w) \geq \#_a(w) + 2$ and $|w|$ is even ($w \in \{a, b\}^+$),
- $\phi(w; L(G_0)) = 0$ if and only if either $w = \lambda$ or $|w|$ is odd ($w \in \{a, b\}^*$).

So the fuzzy context-free grammar G_0 describes the set of all nonempty even length strings over $\{a, b\}$ with preferably as many a's as b's (degree of membership equal to 1). Occasionally, some a's in these nonempty even length strings may be changed into b's or vice versa; the former happens to be a quite less severe incident than the latter (degrees of membership 0.9 and 0.1, respectively). □

3. Generalized Fuzzy Context-Free Grammars

In this section we address the question how tiny mistakes and big blunders can be described within the framework of fuzzy context-free grammars and their generalizations. Our main result determines the expressive power of these generalized fuzzy context-free grammars; cf. Theorem 3.3 and Corollary 3.4.

To be more concrete, let us return to Example 2.2. The principal aim of the fuzzy context-free grammar G_0 is to generate the (crisp) language $L(G_1)$. Applying the rule $S \to BB$ instead of either $S \to AB$ or $S \to BA$ one or more times during a derivation, results in a terminal string w that satisfies: $\#_b(w) \geq \#_a(w) + 2$, $|w|$ is even, and $\phi(w; L(G_0)) = 0.9$. So such terminal strings w may be considered as "tiny mistakes". On the other hand, using the rule $S \to AA$ instead of either $S \to AB$ or $S \to BA$ one or more times, yields a w in Σ^* with $\#_a(w) \geq \#_b(w) + 2$, $|w|$ is even, and $\phi(w; L(G_0)) = 0.1$. Strings w of this type may be viewed as "big blunders", since they "hardly belong" to the fuzzy language $L(G_0)$.

Note that P_0 results from P_1 by allowing a finite number of errors. But in general there is an infinite number of ways to perform tasks wrongly. So what happens when we change some $P_1(\alpha)$ into an infinite set, i.e. an infinite language over V? To answer this question we need the notion of language family (Definition 3.1), and a generalization of fuzzy context-free grammars, the so-called fuzzy context-free K-grammars (Definition 3.2).

Definition 3.1. Let Σ_ω be a countably infinite set of symbols. A *family of languages* over Σ_ω is a set of pairs (L, Σ_L) where $L \subseteq \Sigma_L^*$ and Σ_L is a finite subset of Σ_ω. The set Σ_L is assumed to be the minimal alphabet of L. A family K is called *nontrivial* if K contains a language L with $L \cap \Sigma_\omega^+ \neq \varnothing$.

Similarly, 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. Again we assume that Σ_L is minimal with respect to L, i.e., $a \in \Sigma_L$ if and only if the symbol a occurs in a word x with $\phi(x; L) \neq 0$. A family of fuzzy languages K is called *nontrivial* if K contains a language L such that $\phi(x; L) \neq 0$ for some $x \in \Sigma_\omega^+$.

For each family K of fuzzy languages, we define $c(K) = \{c(L) \mid L \in K\}$. □

Usually, we write L instead of (L, Σ_L) for members of a family of (fuzzy) languages. And henceforth, we assume that each family of (fuzzy) languages is closed under isomorphism ("renaming of symbols"). Thus for each family K we assume that for each language L in K over some alphabet Σ and for each bijective mapping $i : \Sigma \to \Sigma_1$ —extended to words and to languages in the usual way— we have $i(L) \in K$.

Examples of simple, nontrivial families of (crisp) languages, which we will need in the sequel, are $\mathrm{SYMBOL} = \{\{\alpha\} \mid \alpha \in \Sigma_\omega\}$, and $\mathrm{FIN} = \{\{w_1, w_2, \cdots, w_n\} \mid w_i \in \Sigma_\omega^*,\ 1 \leq i \leq n,\ n \geq 0\}$. When discussed in the context of fuzzy languages, we assume that for these families we have $\phi(\alpha; \{\alpha\}) = 1$ and $\phi(w_i; \{w_1, \cdots, w_n\}) = 1$ with $1 \leq i \leq n$. The family of finite fuzzy languages will be denoted by FIN_f. Then $c(\mathrm{FIN}_f) = \mathrm{FIN}$.

Definition 3.2. Let K be a family of fuzzy languages. A *fuzzy context-free K-grammar* $G = (V, \Sigma, P, S)$ consists of

- a finite set V of symbols (the *alphabet* of G);
- a finite set Σ of symbols with $\Sigma \subseteq V$ (the *terminal alphabet* of G);
- a special nonterminal symbol S (the *initial* or *start symbol* of G);
- 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 $\phi(\alpha; P(\alpha)) = 1$.

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., $c(A_f(K)) = \{c(L) \mid L \in A_f(K)\}$. □

For the definition of $P^*(S)$ we refer to §2. The mapping P may be called a *nested fuzzy K-substitution* and, similarly, P^* an *iterated nested fuzzy K-substitution*; cf. the corresponding non-fuzzy notions in [6, 12, 2, 3].

Replacing the family K of fuzzy languages in Definition 3.2 by a family of (ordinary, crisp) languages results in the definition of *context-free K-grammar* [12, 2]; for the corresponding family of languages $A(K)$ it is straightforward to show that $A(c(K)) = c(A_f(K))$. For K equal to the family of finite fuzzy languages we obtain: $A(\mathrm{FIN}) = A(c(\mathrm{FIN}_f)) = c(A_f(\mathrm{FIN}_f)) = \mathrm{CF}$ (the family of context-free languages), and $A_f(\mathrm{FIN}_f) = \mathrm{CF}_f$ (the family of fuzzy context-free languages).

Comparing Definitions 2.1 and 3.1 shows that we removed the requirements (ii) and (iii) in 2.1 to obtain 3.1. But (iii) is just a minor point, since we assumed that all the language families involved are closed under isomorphism. Now we turn to the main result of this section which is concerned with removing (ii).

Theorem 3.3. Let K be a family of fuzzy languages that is closed under union with SYMBOL-languages. If $K \supseteq$ SYMBOL, then $A_f(A_f(K)) = A_f(K)$.

Proof: First, we show that $A_f(A_f(K)) \supseteq A_f(K)$. So let L_0 be a language in $A_f(K)$, i.e. there exist a fuzzy context-free K-grammar $G = (V, \Sigma, P, S)$ with $L(G) = L_0$. Consider the 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 $P_0(\alpha) = \{\alpha\}$ for all α in Σ. Then $L(G_0) = L(G) = L_0$, and for each x in Σ^*, we have $\phi(x; L(G_0)) = \phi(x; L(G)) = \phi(x; L_0)$.

Conversely, 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 $L(G_\alpha) = P(\alpha)$. Clearly, we may assume that all nonterminal alphabets $V_\alpha - V$ are mutually disjoint. Thus we have to show that $L(G) \in A_f(K)$. To this end we perform the following steps.

(1) 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. Since K is closed under isomorphism, we introduce a specific new nonterminal symbol A_β with $P(A_\beta) = \{A_\beta, \beta\}$ for each β in Σ and replace β by A_β everywhere else by means of the isomorphism $i(\beta) = A_\beta$.

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

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

with $V_1 = \bigcup\{V_\alpha \mid \alpha \in V\}$.

Now we have that $L(G) = \{Q_\alpha \mid \alpha \in V\}^* \cap \Sigma^*$, and it remains to reduce the finite set $\{Q_\alpha \mid \alpha \in V\}$ of nested fuzzy K-substitutions over V_1 to a single nested fuzzy K-substitution.

(3) Consider the fuzzy context-free K-grammar $G_0 = (V_0, \Sigma, P_0, S_0)$ defined in the following way.

- Assume that the alphabet V consists of n symbols. Define n isomorphisms i_k ($1 \le k \le n$) on the alphabet V_1. We assume that the alphabets $i_k(V_1)$ ($1 \le k \le n$) are mutually disjoint. Then we define the alphabet V_0 of G_0 by $V_0 = V_1 \cup \bigcup\{i_k(V_1) \mid 1 \le k \le n\}$.
- $S_0 = S_S$. Note that $S_S \in V_S$, $V_S \subseteq V_1$, and hence $S_0 \in V_0$.
- The nested fuzzy K-substitution P_0 over V_0 is defined by

$$\begin{array}{lll} P_0(\beta) = \{\beta, i_1(\beta)\} & \text{iff} & \beta \in V_1, \\ P_0(\beta) = \{\beta, i_{k+1}(\alpha)\} \cup Q_\alpha & \text{iff} & \beta \in i_k(V_1),\ \alpha = i_k^{-1}(\beta) \text{ and } 1 \le k < n, \\ P_0(\beta) = \{\beta\} \cup Q_\alpha & \text{iff} & \beta \in i_n(V_1) \text{ and } \alpha = i_n^{-1}(\beta). \end{array}$$

Finally, it is tedious but straightforward to verify that for each string x in Σ^* we have $\phi(x; L(G_0)) = \phi(x; L(G))$. Consequently, $L(G_0) = L(G)$, and hence the fuzzy language $L(G)$ belongs to the family $A_f(K)$, i.e., $A_f(A_f(K)) \subseteq A_f(K)$. □

Corollary 3.4. $A_f(A_f(\mathrm{FIN}_f)) = A_f(\mathrm{CF}_f) = A_f(\mathrm{FIN}_f) = \mathrm{CF}_f$.

Proof: By $A_f(\text{FIN}_f) = \text{CF}_f$ and Theorem 3.3 with K equal to FIN_f. □

According to Corollary 3.4 we may extend the sets $P(\alpha)$ $(\alpha \in V)$ in a fuzzy context-free grammar $G = (V, \Sigma, P, S)$ with a countable infinite number, as long as the resulting sets $P(\alpha)$ still constitute fuzzy context-free languages over V. In this sense we are able to model the case of an infinite number of errors.

Example 3.5. Consider the fuzzy context-free CF_f-grammar $G_2 = (V, \Sigma, P_2, S)$ with $N = V - \Sigma = \{S, A, B\}$, $\Sigma = \{a, b\}$, and P_2 is defined by

$$P_2(S) = P_0(S) \cup L_2 \cup L_3 \cup L_4$$
$$P_2(\alpha) = P_0(\alpha) \qquad \text{iff} \quad \alpha \neq S$$

where P_0 is as in Example 2.2; for the languages $L_2 = \{aA^n bB^n \mid n \geq 1\}$, $L_3 = \{aA^n \mid n \geq 2\}$, and $L_4 = \{B^n \mid n \geq 3\}$, we have $\phi(aA^n bB^n; L_2) = 1$ $(n \geq 1)$, $\phi(aA^n; L_3) = 0.1$ $(n \geq 2)$, and $\phi(B^n; L_4) = 0.9$ $(n \geq 3)$. The other degrees of membership are as in Example 2.2. Then G_2 generates the same fuzzy language as the fuzzy context-free grammar G_0 from Example 2.2. □

4. A Robust Version of Cocke–Younger–Kasami's Algorithm

In this section we give a robust version of Cocke–Younger–Kasami's algorithm (or CYK-algorithm for short) for recognizing fuzzy context-free languages; cf. Algorithm 4.2 below. Here and in the next section we use a minimal notion of robustness: we call a parsing or a recognizing algorithm *robust* if it correctly computes the degree of membership of its input with respect to a given fuzzy context-free grammar.

Usually, the CYK-algorithm is presented in terms of nested **for**-loops filling an upper-triangular matrix; cf. [1, 7, 8]. Here we use an alternative functional formulation from [4] which possesses some advantages: 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 [7] and Algorithm 4.1 below.

In this alternative formulation we need two functions f and g. Henceforth, for each set X, $\mathcal{P}(X)$ denotes the power set of X. Given a λ-free context-free grammar in Chomsky normal form $G = (V, \Sigma, P, S)$, these two functions $f : \Sigma^+ \to \mathcal{P}(N^+)$ and $g : \mathcal{P}(N^+) \to \mathcal{P}(N)$ are defined by:

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

$$f(a) = \{A \mid a \in P(A)\} \tag{1}$$

and extended to words over Σ by

$$f(w) = f(a_1)f(a_2)\cdots f(a_n) \quad \text{if} \quad w = a_1 a_2 \cdots a_n \quad (a_k \in \Sigma,\ 1 \leq k \leq n). \tag{2}$$

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

$$g(\omega) = \bigcup \{g(\chi) \otimes g(\eta) \mid \chi, \eta \in N^+,\ \omega = \chi\eta\} \tag{3}$$

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

$$X \otimes Y = \{A \mid BC \in P(A), \text{ with } B \in X \text{ and } C \in Y\}. \tag{4}$$

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

$$g(M) = \bigcup \{g(\omega) \mid \omega \in M\}. \tag{5}$$

The functional version of the CYK-algorithm from [4] now reads as follows.

Algorithm 4.1. 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^+. Compute $g(f(w))$ and determine whether S belongs to $g(f(w))$.

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

Once we have the CYK-algorithm in this functional version it is easy to obtain a modification for recognizing fuzzy context-free languages.

Algorithm 4.2. 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 (1)–(5) in Algorithm 4.1 with

$$\phi(A;f(a)) = \phi(a;P(A)), \tag{1'}$$

$$\phi(A;X\otimes Y) = \min\{\phi(BC;P(A)),\ \phi(B;X),\ \phi(C;Y)\}, \tag{3'}$$

$$\phi(A;g(\omega)) = \sup\{\phi(A;g(\chi)\otimes g(\eta)) \mid \chi,\eta\in N^+,\ \ \omega=\chi\eta\}, \tag{4'}$$

whereas corresponding equalities for (2) and (5) follow from the definitions of concatenation and finite union, respectively; cf. §2. Finally, compute $\phi(S;g(f(w)))$.

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

Example 4.3. Consider the fuzzy context-free grammar of Example 2.2. Applying Algorithm 4.2 yields

$$\phi(abba;L(G_0))=\phi(S;g(f(abba)))=\phi(S;g(ABBA))=$$
$$=\phi(S;g(A)\otimes g(BBA)\cup g(AB)\otimes g(BA)\cup g(ABB)\otimes g(A))=\ \cdots\ =1$$

and

$$\phi(abbb;L(G_0))=\phi(S;g(f(abbb)))=\phi(S;g(ABBB))=$$
$$=\phi(S;g(A)\otimes g(BBB)\cup g(AB)\otimes g(BB)\cup g(ABB)\otimes g(B))=\ \cdots\ =0.9$$
□

5. A Robust Version of a Recursive Descent Recognizer

Both Algorithms 4.1 and 4.2 are bottom-up algorithms for recognizing λ-free (fuzzy) context-free languages. Functional top-down analogues of Algorithm 4.1 have been introduced in [4], from which we quote Definition 5.1 and Algorithm 5.2. Then we give in Algorithm 5.3 a modification of 5.2 which results in a recursive descent recognizer for fuzzy context-free languages.

Definition 5.1. Let $G=(V,\Sigma,P,S)$ be a context-free grammar and $N=V-\Sigma$. The set $T(\Sigma,N)$ of *terms* over (Σ,N) is the smallest set satisfying

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

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

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

Note that for any two sets of terms S_1 and S_2 $(S_1,S_2\subseteq T(\Sigma,N))$ the entity 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 5.2. 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 $\Sigma^*\cup\{\bot\}$ to $\mathcal{P}(T(\Sigma,N))$ defined as follows. (The symbol $\bot$ will be used to denote "undefined".) First, $A(\bot)=\varnothing$ 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

$$A(x) = \{\lambda \mid x \in P(A)\} \qquad (x \in \Sigma) \tag{6}$$

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

$$A(x) = \bigcup \{B(y)C(z) \mid BC \in P(A),\ y, z \in \Sigma^+,\ x = yz\}. \tag{7}$$

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 5.3. 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, $\phi(\lambda; A(\lambda)) = 1$ and $\phi(t; A(\perp)) = 0$ for each t in $\mathcal{P}(T(\Sigma, N))$. Extend (6)–(7) in Algorithm 5.2 with

$$\phi(\lambda; A(x)) = \phi(x; P(A)) \qquad (x \in \Sigma), \tag{6'}$$

$$\phi(B(y)C(z); A(x)) = \phi(BC; P(A)) \quad \text{with} \quad yz = x \quad (y, z \in \Sigma^+). \tag{7'}$$

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

Example 5.4. Applying Algorithm 5.3 to the fuzzy context-free grammar of Example 2.2 results in

$$\begin{aligned}
\phi(aabb; L(G_0)) = \phi(\lambda; S(aabb)) = \\
= \phi(\lambda; & A(aab)B(b) \cup A(aa)B(bb) \cup A(a)B(abb) \cup \\
& B(aab)A(b) \cup B(aa)A(bb) \cup B(a)A(abb) \cup \\
& A(aab)A(b) \cup A(aa)A(bb) \cup A(a)A(abb) \cup \\
& B(aab)B(b) \cup B(aa)B(bb) \cup B(a)B(abb)) = \cdots = 1
\end{aligned}$$

$$\phi(aaba; L(G_0)) = \phi(\lambda; S(aaba)) = \cdots = 0.1$$

$$\phi(abb; L(G_0)) = \phi(\lambda; S(abb)) = \cdots = 0$$ □

A version of Algorithm 5.2 based on Greibach 2-form has also been discussed in [4], but it will not be considered here in any detail or modification.

6. Concluding Remarks

When we want to use Algorithms 4.2 or 5.3 in case of a fuzzy context-free language specified by a fuzzy context-free CF_f-grammar we first have to apply the construction in the proof of Theorem 3.3 to obtain an equivalent fuzzy context-free grammar (or fuzzy context-free FIN_f-grammar). Then after transforming this second grammar into Chomsky normal form, using a main result from [10], we are ready to apply Algorithms 4.2 or 5.3.

In this paper we treated errors in a rather "macroscopic" fashion: the right-hand side of a grammar rule may have been replaced erroneously by quite a different string. For a more "microscopic" or local treatment of errors in context-free and context-sensitive language recognition using fuzzy grammars we refer to [11, 9].

Both this paper and [11, 9] model the production of errors in a limited way. Actually, fractional degrees of membership attached to grammar rules are only passed on to terminal strings in the end. So a more subtle treatment of errors like $\phi(aA^n; L_3) = (10 * n)^{-1}$ for $n \geq 2$ or $\phi(B^n; L_4) = 0.9 * \exp(3 - n)$ with $n \geq 3$, in Example 2.2 —modeling the unlikeliness of wrongly replacing short strings by very long strings— is not possible in the present approach.

Needless to say that there are many aspects of robustness in parsing and recognizing context-free languages which are not touched upon in this paper: correction of errors, the problems over "overgeneration" and "undergeneration", etc.

Acknowledgement. I am indebted to Rieks op den Akker for some 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: 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.
4. 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.
5. G. Gerla: Fuzzy grammars and recursively enumerable fuzzy languages, *Inform. Sci.* **60** (1992) 137-143.
6. S.A. Greibach: Full AFL's and nested iterated substitution, *Inform. Contr.* **16** (1970) 7–35.
7. M.A. Harrison: *Introduction to Formal Language Theory* (1978), Addison-Wesley, Reading, Mass.
8. J.E. Hopcroft & J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation* (1979), Addison-Wesley, Reading, Mass.
9. 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.
10. E.T. Lee & L.A. Zadeh: Note on fuzzy languages, *Inform. Sci.* **1** (1969) 421-434.
11. 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.
12. J. van Leeuwen: A generalization of Parikh's theorem in formal language theory, pp. 17-26 in: in J. Loeckx (ed.): *2nd ICALP*, Lect. Notes in Comp. Sci. **14** (1974), Springer-Verlag, Berlin, etc.