Sādhanā (2018) 43:134
https://doi.org/10.1007/s12046-018-0833-y

Error tolerance for the recognition of faulty strings in a regulated grammar using fuzzy sets

AJAY KUMAR*, NIDHI KALRA and SUNITA GARHWAL

Computer Science and Engineering Department, Thapar Institute of Engineering & Technology, Patiala 147004, India
e-mail: ajayloura@gmail.com

MS received 25 July 2017; revised 5 January 2018; accepted 8 January 2018; published online 10 July 2018

Abstract. To overcome the limitations of context-free and context-sensitive grammars, regulated grammars have been proposed. In this paper, an algorithm is proposed for the recognition of faulty strings in regulated grammar. Furthermore, depending on the errors and certainty, it is decided whether the string belongs to the language or not based on string membership value. The time complexity of the proposed algorithm is $O(|G_R^2|\cdot|w|)$, where $|G_R|$ represents the number of production rules and $|w|$ is the length of the input string, w. The reader is provided with numerical examples by applying the algorithm to regularly controlled and matrix grammar. Finally, the proposed algorithm is applied in the Hindi language for the recognition of faulty strings in regulated grammar as a real-life application.

Keywords. Fuzzy regulated grammar; fuzzy sets; Chomsky normal form; regularly controlled grammar; matrix grammar.

1. Introduction

Context-free grammars are not able to cover all aspects of linguistic phenomena. The membership of a string in a context-sensitive grammar is decided in exponential time. To overcome the limitations of context-free and context-sensitive grammars, the concept of regulated grammar [1–3] was introduced. Regulated grammar is a class of grammar in which certain restrictions are imposed on context-free productions, making them more powerful than context-free grammar. Regulated grammar is classified into context and rule-based grammar. In context-based regulated grammar, restrictions are imposed on the context, whereas in rule-based regulated grammar, the restrictions are imposed on the application of rules.

In formal languages, the input string is either accepted ($\mu(w)=1$) or rejected ($\mu(w)=0$), where μ denotes the membership of a string, w. To reduce the gap between natural and classical formal languages, fuzziness has been introduced into formal languages, thereby giving rise to fuzzy languages. In recent years, fuzzy languages have been applied in various areas, such as intelligent interface design [4, 5], clinical monitoring [6], neural networks [7, 8] and pattern recognition [9]. Various fuzzy inference methods [10–12] are used in the design of fuzzy systems.

Motivated by the prior work of recognition of imperfect strings in context-free grammar [13] and context-sensitive grammar [14], in this paper, the recognition of faulty strings generated by the regulated grammar is described. As well, the problem of single and multiple fault occurrences is discussed. Faulty string refers to some imperfection caused in the string. Single fault refers to imperfection at one position in a string, whereas multiple faults refer to imperfection at more than one position in a string. Here, fuzzy sets are used to compute the membership of strings that do not exactly match with the grammar.

1.1 *Previous work*

Regularly controlled and matrix grammar were proposed by Ginsburg *et al* [15] and Cremers *et al* [16], respectively. In regularly controlled grammar, the control is regulated by the sequence of productions in the regular expression. In matrix grammar, control is regulated by following the same sequence given in the matrix. Solar and Meduna [17] proposed a state translation scheme, which is an extended version of state grammar [18], for producing two output strings in a single derivation. Meduna and Zemek [19] introduced one-sided random context grammars, a variant of random context grammar [20]. Furthermore, Meduna and Zemek [21] introduced one-sided random context

*For correspondence

grammars, with a limited number of right random context rules. Meduna [22] put forth generalized forbidding grammar. Furthermore, Meduna and Zemek [23] also introduced one-sided forbidding grammars, where each rule is further divided into left and right forbidden rules, and each rule consists of a finite set of forbidden strings. Meduna and Zemek [24] established that the power of one-sided forbidding grammars and selective substitution grammars [25, 26] are equivalent in the case of erasing rule grammar. Kalra and Kumar [27, 28] introduced the concept of deterministic deep pushdown automata and fuzziness in deep pushdown automata. Garhwal and Jiwari [29] introduced the concept of fuzziness in parallel regular expression.

The concept of fuzzy languages, a fuzzy subset of strings over an alphabet, was introduced by Lee and Zadeh [30]. Asveld [31] proposed fuzzy context-free K-grammar for describing correct and erroneous sentences. Asveld [32] devised the modified Cocke–Younger–Kasami (CYK) algorithm for recognizing fuzzy context-free grammar. Additionally, Schinder *et al* [13] described an approach for recognizing imperfect strings generated by the context-free grammar. Inui *et al* [14] expanded on the work of Schinder *et al* [13], describing an approach for identifying imperfect strings generated by the context-sensitive grammar. Schinder *et al* [13] and Inui *et al* [14] applied the concept of fuzzy sets to determine the membership values of strings. Here, the concept of errors and fuzzy sets in this context is utilized which was earlier used by Schinder *et al* [13] and Inui *et al* [14] regarding context-free grammar and context-sensitive grammar. Recently, Zhanga *et al* [33] and Bag *et al* [34] have applied the concept of pattern recognition in intelligent computing systems and prediction of consumer intention.

After introducing preliminary concepts in section 2, the following topics have been explained.

- Recognition of faulty strings generated by regulated grammar using the different fuzzy sets has been proposed in section 3.
- The problem of occurrence of multiple faults has been solved by converting the regulated grammar into Chomsky normal form in section 3.
- The complete procedure is depicted by numerical example in section 4.
- The proposed algorithm has been applied on the Hindi language to demonstrate its real- life application in section 5 and the conclusions are provided in section 6.

2. Preliminaries

Let T be an alphabet, λ denotes empty string and $\mu(w)$ denotes the membership of a string such that $0 \leq \mu(w) \leq 1$. _ represents the symbol that can be replaced by any symbol from T.

Definition 2.1 Regular expression over an alphabet T is defined using following rules:

i) ϕ and λ are regular expressions representing the empty set and empty string.
ii) If $a \in T$, then a is a regular expression representing the singleton set $\{a\}$.
iii) If r_1 and r_2 are regular expressions, then $r_1r_2, r_1|r_2$ and r_1* are regular expressions representing concatenation, union, and Kleene closure.

The regulated grammar consists of production rules similar as context-free grammar, but it has a larger generative capacity than context-free grammar. Regulated grammar lies between context-free and context-sensitive grammar. Context-sensitive language (such as ww, $a^nb^nc^n$ and a^{2^n}) involving interleaved dependencies can be represented using regulated grammar.

Definition 2.2 Regulated grammar is $G_R=(G,R)$, where $G=(N,T,S,P)$ is a context-free grammar and R is the restriction applied on the derivations of strings. R depends on the type of regulated grammar [1–3].

Regularly controlled grammar is a rule-based regulated grammar, where the restriction on rules is applied using a regular expression.

Definition 2.3 Regularly controlled grammar [1–3] is $G_{RC}=(G,r)$, where $G=(N,T,S,P)$ is a context-free grammar and r is a regular expression over P for controlling the derivations of strings.

Language L (G_{RC}) represented by regularly controlled grammar G_{RC} consists of all words $w \in T*$ generated by the following sequence:

$$S \overset{pr_1}{\rightarrow} w_1 \overset{pr_2}{\rightarrow} w_2 \cdots \overset{pr_n}{\rightarrow} w \text{ such that } pr_1, pr_2, \ldots, pr_n \in P$$

Example 2.1 Consider the regularly controlled grammar [2] $G_{RC}=(G_1,r)$, where $G_1=(\{S,C,D\},\{c,d\},S,P)$ where the production P is of the following forms:

$$\begin{array}{ll} pr_0 : S \rightarrow CD & pr_1 : C \rightarrow cC \\ pr_2 : D \rightarrow cD & pr_3 : C \rightarrow dC \\ pr_4 : D \rightarrow dD & pr_5 : C \rightarrow c \\ pr_6 : D \rightarrow c & pr_7 : C \rightarrow d \\ pr_8 : D \rightarrow d & \end{array}$$

$$r = pr_0(pr_1pr_2, pr_3pr_4)^*(pr_5pr_6, pr_7pr_8)$$

Consider $w=cdcd$

$$S \overset{pr_0}{\rightarrow} CD \overset{pr_1}{\rightarrow} cCD \overset{pr_2}{\rightarrow} cCcD \overset{pr_7}{\rightarrow} cdcD \overset{pr_8}{\rightarrow} cdcd$$

Hence, $L(G_{RC})=\{ww|w \in (c,d)^+\}$ is a non-context-free language.

Matrix grammar is a rule-based regulated grammar, where restrictions on productions are applied using

matrices. Here, we have to apply all productions of a matrix before starting with another matrix.

Definition 2.4 Matrix grammar [1–3] is $G_M=(G,r_m)$, where $G=(N,T,S,M)$ and $M=(m_1,m_2,\ldots,m_n),n\geq 1$ is a finite set of matrices, where each m_i consists of a finite set of context-free production and r_m is a regular expression over the set of matrices m_i.

Example 2.2 Consider $G_M=(G,r_m)$, where $G=(\{S,C,D\},\{c,d\},S,\{m_0,m_1,m_2,m_3,m_4\})$ be the matrix grammar [2] where

$$\begin{aligned} m_0 &= (p_0 : S \rightarrow CD),\\ m_1 &= (p_1 : C \rightarrow cC, p_2 : D \rightarrow cD),\\ m_2 &= (p_3 : C \rightarrow dC, p_4 : D \rightarrow dD),\\ m_3 &= (p_5 : C \rightarrow c, p_6 : D \rightarrow c),\\ m_4 &= (p_7 : C \rightarrow d, p_8 : D \rightarrow d), \end{aligned}$$

The control set is $r_m=m_0(m_1,m_2)^*(m_3,m_4)$.

Consider $w=dcdc$

$$S \overset{m_0:p_0}{\rightarrow} CD \overset{m_2:p_3}{\rightarrow} dCD \overset{m_2:p_4}{\rightarrow} dCdD \overset{m_3:p_5}{\rightarrow} dcdD \overset{m_3:p_6}{\rightarrow} dcdc$$

Hence, $L(G_M)=\{ww|w\in(c,d)^+\}$ is a non-context-free language.

Definition 2.5 Fuzzy grammar [30] is quadruple $G_f=(N,T,P,S)$, where P is a set of fuzzy productions of the form $\{\alpha \xrightarrow{\mu} \beta | 0 \leq \mu \leq 1 \wedge \alpha,\beta \in (N \cup T)^*\}$.

Definition 2.6 Fuzzy language L [30] is a fuzzy set of T^*·$L=\{(x,\mu(x))|x\in T^*$ and $0\leq\mu(x)\leq 1\}$ where $\mu(x)$ denote the degree of membership of string x.

$\mu(x)$ is obtained by taking a minimum of all derivation rules from S to derive x if a single derivation exists for the string x. If there exists more than one derivation for a particular string x, then max is taken over the minimum of all the derivation rules from starting from S to derive x.

Example 2.3 [30] Consider $L=\{(0,0.8),(00,0.4),(10,0.2),(1,0.2),(11,0.6)\}$. Here L describes a fuzzy language over {0, 1}. Zero membership strings are not represented in L.

Definition 2.7 Fuzzy regulated grammar $G_{fr}(G_R,E,w)$, where

i) G_R is a regulated grammar in Chomsky normal form.
ii) E is the error set of productions constructed from the original productions P using the replace operator (replacing a terminal symbol with another terminal in $p\in P$), add operator (adding an extra terminal in $p\in P$) and remove operator (removing a terminal in $p\in P$). Errors in the input string are classified into following three types:
   - *Replacement Error* [13, 14]: In replacement error, a terminal is substituted with another terminal symbol from T.
   - *Addition Error* [13, 14]: In addition error, an extra new terminal symbol is added from T. This new terminal can be placed before or after the terminal.
   - *Removal Error* [13, 14]: In removal error, some terminal is deleted.
iii) w is a weighted function such that $\{(p,w(p))|p\in P\wedge w(p)\in[0,1]\}$. It is used to assign the degree of membership to the production rules.

3. Proposed algorithm for the recognition of faulty strings in regulated grammar using fuzzy sets

In classical automata theory, a string is either accepted or rejected. Fuzzy languages close the gap between natural and formal languages. In the fuzzy set, the degree to which an element belongs to the set is referred to as the degree of membership [35]. In this section, an algorithm is proposed for the recognition of faulty strings in regulated grammar using fuzzy sets (*RFSRG*). Given a regularly controlled grammar $G_{RC}=(G,r)$. For avoiding multiple errors, in step 1, $G_{RC}=(G,r)$ is converted into $G_{RC'}=(G',r')$ where G' is a Chomsky normal form of G. Every production $p_i \in G$ is converted into productions $p_i^1,p_i^2,\ldots,p_i^k|k\geq 1$ of G'. Specifically, G is converted into Chomsky normal form G' [36].

In step 2, control derivation r of G_{RC} is converted into r' of $G_{RC'}$ such that $L(G',r')=L(G,r)-\{\lambda\}$.

Procedure 1: Regularly controlled_control_derivation $(G_{RC},G_{RC'})$
Input: Regularly controlled grammar $G_{RC}=(G,r)$ and $G_{RC'}=(G',r')$ where G' is Chomsky normal form of G and initially $r'=r$.
Output: Control derivation r' such that $L(G',r')=L(G,r)-\{\lambda\}$.
begin
  do $\exists\, p_i \in r$
    Replace p_i with $p_i^1,p_i^2,\ldots,p_i^k$ in r'.
  od
end

Example 3.1 Consider example 2.1, where the production rules P is of the following forms:

$$\begin{aligned} &pr_0 : S \rightarrow CD \quad pr_1 : C \rightarrow cC\\ &pr_2 : D \rightarrow cD \quad pr_3 : C \rightarrow dC\\ &pr_4 : D \rightarrow dD \quad pr_5 : C \rightarrow c\\ &pr_6 : D \rightarrow c \quad pr_7 : C \rightarrow d\\ &pr_8 : D \rightarrow d \end{aligned}$$

$$r = pr_0(pr_1pr_2,pr_3pr_4)^*(pr_5pr_6,pr_7pr_8)$$

Equivalent Chomsky normal form (CNF) for the grammar G_{RC}:

$$pr_0 : S \rightarrow CD \qquad pr_1^1 : C \rightarrow A_1C$$
$$pr_1^2 : A_1 \rightarrow c \qquad pr_2^1 : D \rightarrow A_2D$$
$$pr_2^2 : A_2 \rightarrow c \qquad pr_3^1 : C \rightarrow A_3C$$
$$pr_3^2 : A_3 \rightarrow d \qquad pr_4^1 : D \rightarrow A_4D$$
$$pr_4^2 : A_4 \rightarrow d \qquad pr_5 : C \rightarrow c$$
$$pr_6 : D \rightarrow c \qquad pr_7 : C \rightarrow d$$
$$pr_8 : D \rightarrow d$$

r is converted into $r'=pr_0(pr_1^1pr_1^2pr_2^1pr_2^2,pr_3^1pr_3^2pr_4^1pr_4^2)^*(pr_5$-$pr_6,pr_7pr_8)$.

In step 3, we apply replace, add and remove operators to the productions of the form $A\rightarrow a|A\in N$ and $a\in T$. Hence, there is no possibility of occurrence of multiple errors using one production. To accumulate the faulty string, replace, add and remove operators are considered. Similar attempts have been made for the recognition of faulty strings in context-free and context-sensitive grammar [13, 14].

Procedure 2: Error_productions $(G_{RC'}, P')$

Input: Regularly controlled grammar $G_{RC'} = (G', r')$ with production set P.

Output: Additional production P' is generated using replace, add and remove operator.

Initialization: $P' = \phi$

begin

do $\exists p_i \in P$

If p_i is of the form $A \rightarrow a \mid A \in N \wedge a \in T$

$P' = P' \cup \{A \rightarrow \lambda\}$ **//Using Remove operator**

do $\exists b \in T$

If $(a \neq b)$

$P' = P' \cup \{A \rightarrow b\}$ **//Using Replace operator**

$P' = P' \cup \{A \rightarrow ba\} \cup \{A \rightarrow ab\}$ **//Using Add operator**

od

od

$P \leftarrow P \cup P'$

end

As regularly controlled grammar is ambiguous, in step 4, different derivations for the string, $w\notin L$, were generated. In one derivation of a string, total $2|w|-1$ productions are used. In each derivation of a string, w, the number of productions of the form $A\rightarrow BC|A,B,C\in N$ can be $|w|-1$ and the number of productions of the form $A\rightarrow a|A\in N\wedge a\in T$ can be $|w|$.

In step 5, a fuzzy set E is used for determining the number of erroneous substitutions. Membership of a faulty string is determined by $\mu_E(w) = \frac{|w|-|E_p|}{|w|}$ where $|w|$ represents the length of the string, w, and $|E_p|$ represents the number of error productions used for the generation of string w.

A certainty factor was defined that can be used for the acceptance and rejection of strings. In step 6, a fuzzy set T_f is used for determining the derivation with minimal production rules utilized to derive the faulty string, w. Membership of a faulty string is determined by:

$\mu_{T_f}(w) = 1 - S$ shaped membership function

$$\mu_{T_f}(w) = 1 - \begin{cases} 0, & x \le a \\ 2\left(\frac{x-a}{b-a}\right)^2, & a < x \le (a+b)/2 \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2, & (a+b)/2 < x < b \\ 1, & x \ge b \end{cases} \qquad (1)$$

where a and b represent the minimum and maximum number of productions used to derive the faulty string, w, in all derivations, respectively. x denotes the actual number of productions used in a particular derivation of the faulty string, w. In this a and b plots the extreme points of the sloped curve, therefore making it S-shape. In step 7, the optimal derivation for the string, w, was chosen. The optimal derivation was found by choosing the highest degree of membership obtained from E-set and T_f-set.

Procedure 3: Matrix_Control_derivation $(G_M, G_{M'})$

Input: Matrix Grammar $G_M = (G, r_m)$ and $G_{M'} = (G', r_m')$

where G' is Chomsky normal form of G, and initially $r_m' = r_m$.

Output: Control derivation rm' such that $L(G', r_m') = L(G, r_m) - \{\lambda\}$.

begin

$r_m' = r_m$

do $\exists (p_i \in m_i \wedge m_i \in r_m)$

Replace entry of $(p_i \in m_i)$ *with* $(p_i^1, p_i^2 ... \in m_i')$ *in* $m_i' \in r_m'$

end

If a matrix grammar G_M=(G,r_m) is given instead of regularly controlled grammar. During, the conversion of G into Chomsky normal form G', every production $p_i \in G$ is

replaced by production $p_i^1, p_i^2, \ldots p_i^k | k \geq 1$ of G'. In step 2 of *RFSRG*, we call procedure Matrix_Control_derivation (G_M, G_M'). The rest of the algorithm remains the same.

Algorithm 3.1: $RFSRG(G_{RC}, w, \mu(w))$

Input: A regularly controlled grammar $G_{RC} = (G, r)$ and a faulty string w.

Output: Degree of membership of the string w.

Method

begin

1. Convert context-free grammar G into Chomsky normal form G'.
2. Regularly controlled_control_derivation $(G_{RC}, G_{RC'})$.
3. Error_productions $(G_{RC'}, P')$.
4. Generate different derivation for the faulty string w using P'.
5. Compute the degree of membership μ_E using the fuzzy set E for each derivation of the string w.
$$\mu_E(w) = \frac{|w| - |E_p|}{|w|}$$
6. Compute degree of membership μ_T using the fuzzy set T_f for each derivation of the string w using equation (1).
7. Optimal derivation of string w:
For each derivation of w
$\sigma_i(w) = \min(\mu_E(w), \mu_{T_f}(w))$ Here $\sigma_i(w)$ denote the minimum degree of membership for i^{th} derivation of a string w using E and T_f sets.
End of loop
$\mu(w) = \max(\sigma_i(w))$

End

Example 3.2 Consider $G_M=(G,r_m)$, where $G=(\{S,C,D\}, \{c,d\}, S, \{m_0,m_1,m_2,m_3,m_4\})$ be the matrix grammar of example 2.2 where

$$m_0 = (p_0 : S \rightarrow CD),$$
$$m_1 = (p_1 : C \rightarrow cC, p_2 : D \rightarrow cD),$$
$$m_2 = (p_3 : C \rightarrow dC, p_4 : D \rightarrow dD),$$
$$m_3 = (p_5 : C \rightarrow c, p_6 : D \rightarrow c),$$
$$m_4 = (p_7 : C \rightarrow d, p_8 : D \rightarrow d),$$

The control set is $r_m = m_0(m_1,m_2)^*(m_3,m_4)$.

Equivalent CNF for the grammar G_M:

$$m_0' = (p_0 : S \rightarrow CD),$$
$$m_1' = (p_1^1 : C \rightarrow A_1C, p_1^2 : A_1 \rightarrow c, p_2^1 : D \rightarrow A_2D, p_2^2 : A_2 \rightarrow c),$$
$$m_2' = (p_3^1 : C \rightarrow A_3C, p_3^2 : A_3 \rightarrow d, p_4^1 : D \rightarrow A_4D, p_4^2 : A_4 \rightarrow d),$$
$$m_3' = (p_5 : C \rightarrow c, p_6 : D \rightarrow c),$$
$$m_4' = (p_7 : C \rightarrow d, p_8 : D \rightarrow d)$$

The control set is r_m' is $m_0'(m_1',m_2')^*(m_3',m_4')$.

Theorem 1 *Given a regulated grammar G_R and string w. The time complexity of the RFSRG algorithm is equal to $O(|G_R^2| \cdot |w|)$, where $|G_R|$ represents the size of productions and $|w|$ represents the length of the string respectively.*

Proof In Step1 of *RFSRG* algorithm, regulated grammar G_R is converted into Chomsky Normal form $G_{R'}$ using DEL, TERM, UNIT and BIN operations [36]. Size of $G_{R'}$ i.e. $|G_{R'}|$ depends on the order in which DEL, BIN, UNIT and TERM are carried out [36]. TERM always give a linear increase in the size of the grammar. If we carry out the transformation in BIN→DEL→UNIT, then $|G_{R'}|=O(|G_R^2|)$. In step 2, the control derivation G_R is converted into $G_{R'}$. Each production $p_i \in G_R$ may be converted into a number of productions say $pr_i^1, pr_i^2, \ldots, pr_i^k | k \geq 1$ in $G_{R'}$ and each substitution of p_i in control derivation r gives the corresponding control derivation r'. Each substitution can be carried out in $O(|1|)$. Therefore, the control derivation r can be converted into r' in $O(|G_R|)$.

In step 3, we add additional productions using replace, add and remove operators. Thus, the number of productions is of the size $O(|G_R^2|)$. For each production, four faulty productions (one each by replace and remove operation and two productions by add operations) are generated. Addition of each faulty productions can be carried out in $O(|1|)$,

thereby giving the overall complexity of the second step is $O(|G_R^2|)$.

In each derivation of a string, w, the number of productions of the form $A \rightarrow BC | A,B,C \in N$ can be $|w|-1$ and the number of productions of the form $A \rightarrow a | A \in N \wedge a \in T$ can be $|w|$. Therefore total $2|w|-1$ productions are used in one derivation of a string. Considering at each step of the derivation, we can choose a production from $O(|G_R^2|)$ productions. Hence the overall complexity is $O(|G_R^2| \cdot |w|)$. Step 5 can be carried out in $O(|1|)$. In step 6, for finding a, b and x values, complexity is $O(|G_R^2|)$ and further computation of $\mu_{T_f}(w)$ can be carried out in $O(|1|)$. So overall complexity of Step 6 is $O(|G_R^2|)$. Step 7 can be carried out in $O(|1|)$. Thus the overall complexity of the algorithm is $O(|G_R^2| \cdot |w|)$.

The proposed algorithm works on all rule-based regulated grammars with little modifications in applications of rules.

4. Numerical examples

In this section, we apply *RFSRG* algorithm to regularly controlled grammar.

Example 4.1 Consider the regularly controlled grammar of example 2.1 $G_{RC}=(G_1,r)$, where $G_1=(\{S,C,D\},\{c,d\},S,P)$ where the production P is of the following forms:

$$\begin{array}{ll} pr_0 : S \rightarrow CD & pr_1 : C \rightarrow cC \\ pr_2 : D \rightarrow cD & pr_3 : C \rightarrow dC \\ pr_4 : D \rightarrow dD & pr_5 : C \rightarrow c \\ pr_6 : D \rightarrow c & pr_7 : C \rightarrow d \\ pr_8 : D \rightarrow d & \end{array}$$

$$r = pr_0(pr_1pr_2, pr_3pr_4)^*(pr_5pr_6, pr_7pr_8)$$

Equivalent Chomsky normal form (CNF) for the grammar G_{RC}:

$$\begin{array}{ll} pr_0 : S \rightarrow CD & pr_1^1 : C \rightarrow A_1C \\ pr_1^2 : A_1 \rightarrow c & pr_2^1 : D \rightarrow A_2D \\ pr_2^2 : A_2 \rightarrow c & pr_3^1 : C \rightarrow A_3C \\ pr_3^2 : A_3 \rightarrow d & pr_4^1 : D \rightarrow A_4D \\ pr_4^2 : A_4 \rightarrow d & pr_5 : C \rightarrow c \\ pr_6 : D \rightarrow c & pr_7 : C \rightarrow d \\ pr_8 : D \rightarrow d & \end{array}$$

r is converted into $r'=pr_0(pr_1^1pr_1^2pr_2^1pr_2^2, pr_3^1pr_3^2pr_4^1pr_4^2)^*(pr_5 pr_6, pr_7pr_8)$.

Using fuzzy replace operator:

$$\begin{array}{ll} pr_9 : A_1 \rightarrow \_ & pr_{10} : A_2 \rightarrow \_ \\ pr_{11} : A_3 \rightarrow \_ & pr_{12} : A_4 \rightarrow \_ \\ pr_{13} : C \rightarrow \_ & pr_{14} : D \rightarrow \_ \\ pr_{15} : C \rightarrow \_ & pr_{16} : D \rightarrow \_ \end{array}$$

Using fuzzy add operator:

$$\begin{array}{ll} pr_{17} : A_1 \rightarrow \_c|c\_ & pr_{18} : A_2 \rightarrow \_c|c\_ \\ pr_{19} : A_3 \rightarrow \_d|d\_ & pr_{20} : A_4 \rightarrow \_d|d\_ \\ pr_{21} : C \rightarrow \_c|c\_ & pr_{22} : D \rightarrow \_c|c\_ \\ pr_{23} : C \rightarrow \_d|d\_ & pr_{24} : D \rightarrow \_d|d\_ \end{array}$$

Using fuzzy remove operator:

$$\begin{array}{ll} pr_{25} : A_1 \rightarrow \lambda & pr_{26} : A_2 \rightarrow \lambda \\ pr_{27} : A_3 \rightarrow \lambda & pr_{28} : A_4 \rightarrow \lambda \\ pr_{29} : C \rightarrow \lambda & pr_{30} : D \rightarrow \lambda \\ pr_{31} : C \rightarrow \lambda & pr_{32} : D \rightarrow \lambda \end{array}$$

Consider the string $w = ddddcd \notin L$. Table 1 depicts some ways for deriving the string w.

Table 2 represents the confidence level for the string w using E-set.

Table 3 represents the confidence level for the string w using T_f-set. Table 4 represents the final interpretations with fuzzy confidence for the string $w = ddddcd$.

The optimal derivation for the string $w = ddddcd$ is as follows:

$$\begin{aligned} \mu(w) &= \max(0,\ 0,\ 0.6667,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.3333) \\ &= 0.6667 \end{aligned}$$

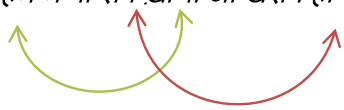
By choosing a certain level of confidence λ_c, we say the string is accepted if $\mu(w) \geq \lambda_c$. Application of RFSRG algorithm to matrix grammar (for details refer "Appendix 1").

5. Real -life application of *RFSRG* algorithm in the Hindi language

Regulated languages are the most appropriate formal method for representing human languages.

In linguistic topology, the Hindi language contains cross-serial dependency, and its structure is similar to the German and Dutch languages. Consider the following sentence of the Hindi Language spoken in India.

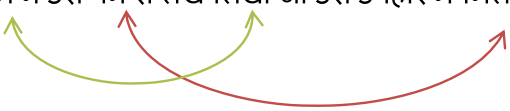
S₁: मोहन ने वो सेब ख़ाया जो खराब हो गया।



Meaning: Mohan that apple ate which rotten was.

English: Mohan ate that apple which was rotten.

S₂: राम ने उस पेन से लेख लिखा जो उसे उपहार मे मिला था।



Meaning: Ram has that pen with essay written which him gifted was

English: Ram has written the essay with that pen which was gifted to him.

Table 1. Various derivation for the string w=*ddddcd*.

Derivation 1	$S \rightarrow_{p0} CD \rightarrow_{p_3^1} A_3CD \rightarrow_{p_3^2} dCD \rightarrow_{p_4^1} dCA_4D \rightarrow_{p_4^2} dCdD \rightarrow_{p_3^1} dA_3CdD \rightarrow_{p19} dddCdD \rightarrow_{p4'} dddCdA_4D \rightarrow_{p20} dddCdcdD \rightarrow_{p29} ddddcdD \rightarrow_{p30} ddddcd$
Derivation 2	$S \rightarrow_{p0} CD \rightarrow_{p_3^1} A_3CD \rightarrow_{p_3^2} dCD \rightarrow_{p_4^1} dCA_4D \rightarrow_{p_4^2} dCdD \rightarrow_{p_1^1} dA_1CdD \rightarrow_{p9} ddCdD \rightarrow_{p_2^1} ddCdA_2D \rightarrow_{p_2^2} ddCdcD \rightarrow_{p7} ddddcD \rightarrow_{p8} ddddcd$
Derivation 3	$S \rightarrow_{p0} CD \rightarrow_{p_3^1} A_3CD \rightarrow_{p19} ddCD \rightarrow_{p_4^1} ddCA_4D \rightarrow_{p20} ddCdcD \rightarrow_{7} ddddcD \rightarrow_{8} ddddcd$
Derivation 4	$S \rightarrow_{p0} CD \rightarrow_{p_3^1} A_3CD \rightarrow_{p_3^2} dCD \rightarrow_{p_4^1} dCA_4D \rightarrow_{p_4^2} dCdD \rightarrow_{p_3^1} dA_3CdD \rightarrow_{p_3^2} ddCdD \rightarrow_{p_4^1} ddCdA_4D \rightarrow_{p12} ddCdcD \rightarrow_{p7} ddddcD \rightarrow_{p8} ddddcd$
Derivation 5	$S \rightarrow_{p0} CD \rightarrow_{p_1^1} A_1CD \rightarrow_{p9} dCD \rightarrow_{p_2^1} dCA_2D \rightarrow_{p10} dCdD \rightarrow_{p_1^1} dA_1CdD \rightarrow_{p9} ddCdD \rightarrow_{p_2^1} ddCdA_2D \rightarrow_{p_2^2} ddCdcD \rightarrow_{p7} ddddcD \rightarrow_{p8} ddddcd$
Derivation 6	$S \rightarrow_{p0} CD \rightarrow_{p_1^1} A_1CD \rightarrow_{p9} dCD \rightarrow_{p_2^1} dCA_2D \rightarrow_{p10} dCdD \rightarrow_{p_3^1} dA_3CdD \rightarrow_{p19} dddCdD \rightarrow_{p_4^1} dddCdA_4D \rightarrow_{p20} dddCdcdD \rightarrow_{p29} ddddcdD \rightarrow_{p30} ddddcd$
Derivation 7	$S \rightarrow_{p0} CD \rightarrow_{p_1^1} A_1CD \rightarrow_{p9} dCD \rightarrow_{p_2^1} dCA_2D \rightarrow_{p10} dCdD \rightarrow_{p_3^1} dA_3CdD \rightarrow_{p_3^2} ddCdD \rightarrow_{p_4^1} ddCdA_4D \rightarrow_{p12} ddCdcD \rightarrow_{p7} ddddcD \rightarrow_{p8} ddddcd$
Derivation 8	$S \rightarrow_{p0} CD \rightarrow_{p_3^1} A_3CD \rightarrow_{p_3^2} dCD \rightarrow_{p_4^1} dCA_4D \rightarrow_{p_4^2} dCdD \rightarrow_{p_3^1} dA_3CdD \rightarrow_{p_3^2} ddCdD \rightarrow_{p_4^1} ddCdA_4D \rightarrow_{p12} ddCdcD \rightarrow_{p13} ddddcD \rightarrow_{p14} ddddcd$
Derivation 9	$S \rightarrow_{p0} CD \rightarrow_{p_3^1} A_3CD \rightarrow_{p19} ddCD \rightarrow_{p_4^1} ddCA_4D \rightarrow_{p20} ddCdcD \rightarrow_{13} ddddcD \rightarrow_{14} ddddcd$

Table 2. Confidence level for the string w using E-set.

Derivation no.	1	2	3	4	5	6	7	8	9
Grade of membership	0.3333	0.8333	0.6667	0.8333	0.5	0	0.5	0.5	0.3333

Table 3. Confidence level for the string w using T_f-set.

Derivation no.	1	2	3	4	5	6	7	8	9
Grade of membership	1–1=0	1–1=0	1–0=1	1–1=0	1–1=0	1–1=0	1–1=0	1–1=0	1–0=1

Table 4. Final interpretations with fuzzy confidence for the string w=*ddddcd*.

Derivation no.	1	2	3	4	5	6	7	8	9
$\mu(w \in E)$	0.3333	0.8333	0.6667	0.8333	0.5	0	0.5	0.5	0.3333
$\mu(w \in T_f)$	0	0	1	0	0	1	0	0	1
Min	0	0	0.6667	0	0	0	0	0	0.3333

Table 5. Various derivation for the string w=*cdcdd*.

Derivation 1	Derivation 2	Derivation 3	Derivation 4
pr_0:$S \rightarrow CD$	pr_0:$S \rightarrow CD$	pr_0:$S \rightarrow CD$	pr_0:$S \rightarrow CD$
pr_1^1:$S \rightarrow A_1CD$	pr_1^1:$S \rightarrow A_1CD$	pr_3^1:$S \rightarrow A_3CD$	pr_3^1:$S \rightarrow A_3CD$
pr_1^2:$S \rightarrow cCD$	pr_1^2:$S \rightarrow cCD$	pr_{11}:$S \rightarrow cCD$	pr_{19}:$S \rightarrow cdCD$
pr_2^1:$S \rightarrow cCA_2D$	pr_2^1:$S \rightarrow cCA_2D$	pr_4^1:$S \rightarrow cCA_4D$	pr_4^1:$S \rightarrow cdCA_4D$
pr_{18}:$S \rightarrow cCcdD$	pr_2^2:$S \rightarrow cCcD$	pr_4^2:$S \rightarrow cCdD$	pr_4^2:$S \rightarrow cdCdD$
pr_7:$S \rightarrow cdcdD$	pr_7:$S \rightarrow cdcD$	pr_{23}:$S \rightarrow cdcdD$	pr_5:$S \rightarrow cdcdD$
pr_8:$S \rightarrow cdcdd$	pr_{24}:$S \rightarrow cdcdd$	pr_8:$S \rightarrow cdcdd$	pr_{14}:$S \rightarrow cdcdd$

Table 6. Confidence level for the string w using E-set.

Derivation no.	Derivation 1	Derivation 2	Derivation 3	Derivation 4
Grade of membership	0.75	0.75	0.5	0.5

Table 7. Confidence level for the string w using T_f-set.

Derivation no.	Derivation 1	Derivation 2	Derivation 3	Derivation 4
Grade of membership	1–0=1	1–0=1	1–0=1	1–0=1

Table 8. Final interpretations with fuzzy confidence for the string w=*cdcdd*.

Derivation No.	Derivation 1	Derivation 2	Derivation 3	Derivation 4
$\mu(w \in E)$	0.75	0.75	0.5	0.5
$\mu(w \in T_f)$	1	1	1	1
Min	0.75	0.75	0.5	0.5

In the above sentences, sequential noun phrases *मोहन* (Mohan), *सेब* (apple), राम (Ram), पेन (pen) and the verb phrases *खाया* (ate), *खराब हो गया* (rotten), लिखा (written), उपहार मे मिला था (gifted) both form two separate series of constituents. Let m' be the morphism which maps to *मोहन* and *खाया* to c, *सेब* and *खराब हो गया* to d and all other words of the sentence to the empty string. Therefore, S_1:*cdcd*. Therefore, sentence S_1 is of the form ww, which is a non-context-free language.

Consider m'' be the morphism, which maps राम, पेन, लिखा, उपहार मे मिला था to a,b,c and d, respectively and the dependency relation between राम (Ram) and लिखा (written) is denoted by m and पेन (pen) and उपहार मे मिला था (gifted) by n and other words of the sentence by empty string. Therefore, sentence S_2 is of the form $a^m b^n c^m d^n$, which is a non-context-free language. Cross-serial dependency of the Hindi language can be represented by regulated grammar.

Now consider the sentence S_1 and If the users unintentionally write the S_1 sentence as

S_1': मोहन ने वो सेब ख़ाया जो सेब खराब हो गया।

In this sentence, the user makes an insertion error of one noun phrase सेब (apple). Now the sentence S_1' maps to *cdcdd* instead of *cdcd*. In formal grammar, this string is completely rejected. However, in case of fuzzy regulated grammar, we cannot completely reject the string w. By considering the grammar of example 4.1, Erroneous sentence acceptance S_1' is shown using proposed algorithm

RFSRG. Table 5 depicts few ways for deriving the string w =*cdcdd*.

Table 6 represents the confidence level for the string w using E-set.

Table 7 represents the confidence level for the string w using T_f-set. Table 8 represents the final interpretations with fuzzy confidence for the string $w = cdcdd$.

The optimal derivation for the string $w = cdcdd$ is as follows:

$$\mu(w) = \max(0.75;\ 0.75;\ 0.5;\ 0.5) = 0.75$$

By choosing a certain level of confidence λ_c, we say the string is accepted if $\mu(w) \geq \lambda_c$.

6. Conclusions

In this paper, an algorithm for deriving a string, $w \notin L$, with a certain level of confidence is described. Regulated grammar is converted to Chomsky normal form to avoid multiple error elimination. Furthermore, the algorithm is applied to regularly controlled and matrix grammar. The time complexity of the proposed algorithm is $O(|G_R^2| \cdot |w|)$. By way of numerical examples for a string, $w \notin L$, the determination of a confidence level for the string, w, is also demonstrated Further, the real-life application of a proposed algorithm for the recognition of faulty strings for the natural language, Hindi, is exhibited.

Acknowledgements

One of the authors, Nidhi Kalra was supported under Visvesvaraya PhD Scheme Fellowship by Ministry of Electronics and Information Technology, Government of India.

Appendix 1: Numerical example

In this appendix, the proposed algorithm has been applied to a matrix grammar.

Example 4.2 Consider $G_M = (G, r_m)$, where $G = \{\{S, C, D\}, \{c, d\}, S, \{m_0, m_1, m_2, m_3, m_4\}\}$ be the matrix grammar of example 2.2 where

$$\begin{aligned} m_0 &= (p_0 : S \to CD),\\ m_1 &= (p_1 : C \to cC, p_2 : D \to cD),\\ m_2 &= (p_3 : C \to dC, p_4 : D \to dD),\\ m_3 &= (p_5 : C \to c, p_6 : D \to c),\\ m_4 &= (p_7 : C \to d, p_8 : D \to d), \end{aligned}$$

The control set is $r_m = m_0(m_1, m_2) * (m_3, m_4)$.

Equivalent CNF for the grammar G_M:

$$\begin{aligned} m_0' &= (p_0 : S \to CD),\\ m_1' &= (p_1^1 : C \to A_1C, p_1^2 : A_1 \to c, p_2^1 : D \to A_2D,\\ &\quad p_2^2 : A_2 \to c),\\ m_2' &= (p_3^1 : C \to A_3C, p_3^2 : A_3 \to d, p_4^1 : D \to A_4D,\\ &\quad p_4^2 : A_4 \to d),\\ m_3' &= (p_5 : C \to c, p_6 : D \to c),\\ m_4' &= (p_7 : C \to d, p_8 : D \to d) \end{aligned}$$

The control set is r_m' is $m_0'(m_1', m_2') * (m_3', m_4')$.
Using fuzzy replace operator:

$$\begin{array}{ll} pr_9 : A_1 \to \_ & pr_{10} : A_2 \to \_\\ pr_{11} : A_3 \to \_ & pr_{12} : A_4 \to \_\\ pr_{13} : C \to \_ & pr_{14} : D \to \_\\ pr_{15} : C \to \_ & pr_{16} : D \to \_ \end{array}$$

Using fuzzy add operator:

$$\begin{array}{ll} pr_{17} : A_1 \to \_c|c\_ & pr_{18} : A_2 \to \_c|c\_\\ pr_{19} : A_3 \to \_d|d\_ & pr_{20} : A_4 \to \_d|d\_\\ pr_{21} : C \to \_c|c\_ & pr_{22} : D \to \_c|c\\ pr_{23} : C \to \_d|d\_ & pr_{24} : D \to \_d|d\_ \end{array}$$

Using fuzzy remove operator:

$$\begin{array}{ll} pr_{25} : A_1 \to \lambda & pr_{26} : A_2 \to \lambda\\ pr_{27} : A_3 \to \lambda & pr_{28} : A_4 \to \lambda\\ pr_{29} : C \to \lambda & pr_{30} : D \to \lambda\\ pr_{31} : C \to \lambda & pr_{32} : D \to \lambda \end{array}$$

On considering string $w = ddcd \notin L$. Table A1 depicts few ways for deriving the string $w = ddcd$.

Table A2 represents the confidence level for the string $w$using E-set.

Table A3 and A4 represent the confidence level for the string $w$using T_f-set and final interpretations with fuzzy confidence for the string $w = cdcdcccd$ respectively.

Table A1. Various derivation for the string $w = cdcdcccd$.

Derivation 1	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_1^1]{m_1} A_1CD \xrightarrow[p_1^2]{m_1} cCD \xrightarrow[p_2^1]{m_1} cCA_2D \xrightarrow[p_2^2]{m_1} cCcD \xrightarrow[p_3^1]{m_2} cA_3CcD \xrightarrow[p_3^2]{m_2} cdCcD \xrightarrow[p_4^1]{m_2} cdCcA_4D \xrightarrow[p14]{m_2} cdCccD \xrightarrow[p_1^1]{m_1} cdA_1CccD \xrightarrow[p_1^2]{m_1} cdcCccD \xrightarrow[p_2^1]{m_1} cdcCccA_2D \xrightarrow[p_2^2]{m_1} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 2	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_1^1]{m_1} A_1CD \xrightarrow[p_1^2]{m_1} cCD \xrightarrow[p_2^1]{m_1} cCA_2D \xrightarrow[p_2^2]{m_1} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p9]{m_1} ccCcD \xrightarrow[p_2^1]{m_1} cdCcA_2D \xrightarrow[p_2^2]{m_1} cdCccD \xrightarrow[p_1^1]{m_1} cdA_1CccD \xrightarrow[p_1^2]{m_1} cdcCccD \xrightarrow[p_2^1]{m_1} cdcCccA_2D \xrightarrow[p_2^2]{m_1} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 3	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_1^1]{m_1} A_1CD \xrightarrow[p_1^2]{m_1} cCD \xrightarrow[p_2^1]{m_1} cCA_2D \xrightarrow[p_2^2]{m_1} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p17]{m_1} cdcCcD \xrightarrow[p_2^1]{m_1} cdcCcA_2D \xrightarrow[p18]{m_1} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 4	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_1^1]{m_1} A_1CD \xrightarrow[p_1^2]{m_1} cCD \xrightarrow[p_2^1]{m_1} cCA_2D \xrightarrow[p_2^2]{m_1} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p17]{m_1} cdcCcD \xrightarrow[p_2^1]{m_1} cdcCcA_2D \xrightarrow[p18]{m_1} cdcCcccD \xrightarrow[p_3^1]{m_2} cdcA_3CcccD \xrightarrow[p_3^2]{m_2} cdcdCcccD \xrightarrow[p_4^1]{m_2} cdcdCcccA_4D \xrightarrow[p_4^2]{m_2} cdcdCcccdD \xrightarrow[p31]{m_4} cdcdcccdD \xrightarrow[p32]{m_4} cdcdcccd$
Derivation 5	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_3^1]{m_2} A_3CD \xrightarrow[p11]{m_2} cCD \xrightarrow[p_4^1]{m_2} cCA_4D \xrightarrow[p_{12}]{m_2} cCcD \xrightarrow[p_3^1]{m_2} cA_3CcD \xrightarrow[p_3^2]{m_2} cdCcD \xrightarrow[p_4^1]{m_2} cdCcA_4D \xrightarrow[p12]{m_2} cdCccD \xrightarrow[p_1^1]{m_1} cdA_1CccD \xrightarrow[p_1^2]{m_1} cdcCccD \xrightarrow[p_2^1]{m_1} cdcCccA_2D \xrightarrow[p_2^2]{m_1} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 6	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_3^1]{m_2} A_3CD \xrightarrow[p11]{m_2} cCD \xrightarrow[p_4^1]{m_2} cCA_4D \xrightarrow[p_{12}]{m_2} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p9]{m_1} ccCcD \xrightarrow[p_2^1]{m_1} cdCcA_2D \xrightarrow[p_2^2]{m_1} cdCccD \xrightarrow[p_1^1]{m_1} cdA_1CccD \xrightarrow[p_1^2]{m_1} cdcCccD \xrightarrow[p_2^1]{m_1} cdcCccA_2D \xrightarrow[p_2^2]{m_1} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 7	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_3^1]{m_2} A_3CD \xrightarrow[p11]{m_2} cCD \xrightarrow[p_4^1]{m_2} cCA_4D \xrightarrow[p_{12}]{m_2} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p17]{m_1} cdcCcD \xrightarrow[p_2^1]{m_1} cdcCcA_2D \xrightarrow[p18]{m_1} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 8	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_3^1]{m_2} A_3CD \xrightarrow[p11]{m_2} cCD \xrightarrow[p_4^1]{m_2} cCA_4D \xrightarrow[p_{12}]{m_2} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p17]{m_1} cdcCcD \xrightarrow[p_2^1]{m_1} cdcCcA_2D \xrightarrow[p18]{m_1} cdcCcccD \xrightarrow[p_3^1]{m_2} cdcA_3CcccD \xrightarrow[p_3^2]{m_2} cdcdCcccD \xrightarrow[p_4^1]{m_2} cdcdCcccA_4D \xrightarrow[p_4^2]{m_2} cdcdCcccdD \xrightarrow[p31]{m_4} cdcdcccdD \xrightarrow[p32]{m_4} cdcdcccd$
Derivation 9	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_1^1]{m_1} A_1CD \xrightarrow[p_1^2]{m_1} cCD \xrightarrow[p_2^1]{m_1} cCA_2D \xrightarrow[p_2^2]{m_1} cCcD \xrightarrow[p_3^1]{m_2} cA_3CcD \xrightarrow[p_3^2]{m_2} cdCcD \xrightarrow[p_4^1]{m_2} cdCcA_4D \xrightarrow[p12]{m_2} cdCccD \xrightarrow[p_3^1]{m_2} cdA_3CccD \xrightarrow[p11]{m_2} cdcCccD \xrightarrow[p_4^1]{m_2} cdcCccA_4D \xrightarrow[p12]{m_2} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$
Derivation 10	$S \xrightarrow[p_0]{m_0} CD \xrightarrow[p_1^1]{m_1} A_1CD \xrightarrow[p_1^2]{m_1} cCD \xrightarrow[p_2^1]{m_1} cCA_2D \xrightarrow[p_2^2]{m_1} cCcD \xrightarrow[p_1^1]{m_1} cA_1CcD \xrightarrow[p9]{m_1} ccCcD \xrightarrow[p_2^1]{m_1} cdCcA_2D \xrightarrow[p_2^2]{m_1} cdCccD \xrightarrow[p_3^1]{m_2} cdA_3CccD \xrightarrow[p11]{m_2} cdcCccD \xrightarrow[p_4^1]{m_2} cdcCccA_4D \xrightarrow[p12]{m_2} cdcCcccD \xrightarrow[p7]{m_4} cdcdcccD \xrightarrow[p8]{m_4} cdcdcccd$

Table A2. Confidence level for the string w using E-set.

Derivation no.	1	2	3	4	5	6	7	8	9	10
Grade of Membership	0.875	0.875	0.75	0.5	0.625	0.625	0.5	0.25	0.625	0.625

Table A3. Confidence level for the string w using T_f-set.

Derivation no.	1	2	3	4	5	6	7	8	9	10
Grade of Membership	1-1 = 0	1-1 =0	1-0 =1	1-1 =0	1-1 = 0	1-1 = 0	1-0 = 1	1-1 = 0	1-1 = 0	1-1 =0

Table A4. Final interpretations with fuzzy confidence for the string $w = cdcdcccd$.

Derivation no.	1	2	3	4	5	6	7	8	9	10
$\mu(w \in E)$	0.875	0.875	0.75	0.5	0.625	0.625	0.5	0.25	0.625	0.625
$\mu(w \in T_f)$	1-1=0	1-1=0	1-0=1	1-1=0	1-1=0	1-1=0	1-0=1	1-1=0	1-1=0	1-1=0
Min	0	0	0.75	0	0	0	0.5	0	0	0

$$\mu(w) = \max(0,\ 0,\ 0.75,\ 0,\ 0,\ 0,\ 0.5,\ 0,\ 0,\ 0) = 0.75$$

By choosing a certain level of confidence λ_c, we say string is accepted if $\mu(w) \geq \lambda_c$.

References

[1] Dassow J and Păun G 1989 *Regulated rewriting in Formal Language Theory* (1st ed)., Berlin, Heidelberg: Springer

[2] Dassow J 2004 *Grammars with regulated rewriting*. In: Martin-vide C, Mitrana V and Paun G (Eds.) *Formal Languages and Applications*. Berlin, Heidelberg: Springer, pp. 249–273

[3] Meduna A and Zemek P 2014 *Regulated Grammars and Automata* (1st ed). New York: Springer

[4] Senay H 1992 Fuzzy command grammars for intelligent interface design. *IEEE Transactions on Systems, Man and Cybernetics* 22(5): 1124–1131

[5] Hopcroft J E, Motwani R and Ullman J D 2006 *Introduction to automata theory, languages and computation* (3rd ed). Boston: Addison-Wesley

[6] Steimann F and Adlassnig K P 1994 Clinical monitoring with fuzzy automata. *Fuzzy Sets and Systems* 61(1): 37–42

[7] Giles C L, Omlin C W and Thornber K K 1999 Equivalence in knowledge representation: automata, recurrent neural networks, and dynamical fuzzy systems. *Proceedings of the IEEE* 87(9):1623–1640

[8] Zadeh L A 2004 A note on web intelligence, world knowledge and fuzzy logic. *Data and Knowledge Engineering* 50 (3): 291–304

[9] DePalma G F and Yau S S 1975 Fractionally fuzzy grammars with application to pattern recognition. In: Zadeh L A, Fu K S, Tanaka K and Shimura M (Eds.) *Fuzzy Sets and their Application to Cognitive and Decision Processes*. New York: Academic Press, pp. 329–351

[10] Qiu D 2007 Automata theory based on quantum logic: Reversibilities and pushdown automata. *Theoretical Computer Science* 386(1–2): 38–56

[11] Bělohlávek R 2002 Determinism and fuzzy automata. *Information Sciences* 143(1–4): 205–209

[12] Ignjatović J, Ćirić M and Bogdanović S 2008 Determinization of fuzzy automata with membership values in complete residuated lattices. *Information Sciences* 178(1): 164–180

[13] Schneider M, Lim H and Shoaff W 1992 The utilization of fuzzy sets in the recognition of imperfect strings. *Fuzzy Sets and Systems* 49(3): 331–337

[14] Inui M, Shoaff W, Fausett L and Schneider M 1994 The recognition of imperfect strings generated by fuzzy context sensitive grammars. *Fuzzy sets and systems* 62(1): 21–29

[15] Ginsburg S and Spanier E H 1968 Control sets on grammars. *Math. Systems Theory* 2(2): 159–177

[16] Cremers A and Mayer O 1973 On matrix languages. *Information and Control* 23(1): 86–96

[17] Solar P 2014 Deep Pushdown Transducers and State Translation Schemes. In: *Proceedings of the 20th Conference STUDENT EEICT*, Brno University of Technology, 24 April, pp. 264–268

[18] Kasai T 1970 An hierarchy between context-free and context-sensitive languages. *Journal of Computer and System Sciences* 4(5): 492–508

[19] Zemek P 2013 One-sided random context grammars: Established results and open problems. In: *Proceedings of the*

19th Conference STUDENT EEICT, Brno University of Technology, 25 April, pp. 222–226

[20] Van der Walt A P J 1970 Random context grammars. In: *Proceedings IFIP Congress.* North-Holland, Amsterdam, pp. 66–68

[21] Meduna A and Zemek P 2014 One-sided random context grammars with a limited number of right random context rules. *Theoretical Computer Science* 516: 127–132

[22] Meduna A 1990 Generalized forbidding grammars. *International Journal of Computer Mathematics* 36(1–2): 31–38

[23] Meduna A and Zemek P 2013 Generalized one-sided forbidding grammars. *International Journal of Computer Mathematics* 90(2): 172–182

[24] Meduna A and Zemek P 2012 One-sided forbidding grammars and selective substitution grammars. *International Journal of Computer Mathematics* 89(5): 586–596

[25] Kleijn H C M 1983 *Selective Substitution Grammars Based on Context-Free Productions*. Ph.D. Thesis, Leiden University, Netherlands

[26] Kleijn H C M 1987 Basic ideas of selective substitution grammars. In: Kelemenova A and Kelemen J (Eds.) *Trends Techniques and Problems in Theoretical Computer Science*. Berlin, Germany: Springer, pp. 75–95

[27] Kalra N and Kumar A 2017 Deterministic Deep Pushdown Transducer and its Parallel Version. *The Computer Journal* 61(1): 63–73

[28] Kalra N and Kumar A 2016 Fuzzy state grammar and fuzzy deep pushdown automaton. *Journal of Intelligent and Fuzzy Systems* 31(1): 249–258

[29] Garhwal S and Jiwari R 2016 Parallel fuzzy regular expression and its conversion to epsilon-free fuzzy automaton. *The Computer Journal* 59(9):1383–1391

[30] Lee E T and Zadeh L A 1969 Note on fuzzy languages. *Information Sciences* 1(4): 421–434

[31] Asveld P R J 2005 Fuzzy context-free languages. Part 1: generalized fuzzy context-free grammars. *Theoretical Computer Science* 347(1): 167–190

[32] Asveld P R J 2005 Fuzzy context-free languages. Part 2: Recognition and parsing algorithms. *Theoretical Computer Science* 347(1): 191–213

[33] Zhanga J, Williams S O and Wang H 2017 Intelligent computing system based on pattern recognition and data mining algorithms. *Sustainable Computing: Informatics and Systems.* https://doi.org/10.1016/j.suscom.2017.10.010

[34] Bag S, Tiwari M K and Chan F T S 2017 Predicting the consumer's purchase intention of durable goods: An attribute-level analysis. *Journal of Business Research*. https://doi.org/10.1016/j.jbusres.2017.11.031

[35] Zadeh L A 1965 Fuzzy sets. *Information and Control* 8(3): 338–353

[36] Lange M and Leiß H 2009 To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm. *Informatica Didactica* 8: 2008–2010