J. Math. Computer Sci., 30 (2023), 89–100

Online: ISSN 2008-949X

Journal of Mathematics and Computer Science

Journal Homepage: www.isr-publications.com/jmcs

Regular string token fuzzy Petri nets

A. John Kaspar, D. K. Sheena Christy*

Department of Mathematics, SRM Institute of Science and Technology, Kattankulathur, 6032 203, India.

Abstract

Fuzzy Petri nets are a type of classical Petri nets designed to deal with imprecise and ambiguous data, that have been widely used to represent fuzzy production rules and fuzzy rule-based reasoning. In this paper, we introduce a new model called string token fuzzy Petri nets to generate fuzzy regular languages. Also, we study the equivalences of fuzzy regular grammar and regular string token fuzzy Petri net and establish some closure properties such as union, catenation, kleene closure, reversal, homomorphism and inverse homomorphism of the languages generated by the regular string token fuzzy Petri nets.

Keywords: Fuzzy Petri net, Petri net languages, string token Petri net, fuzzy languages.

2020 MSC: 68Q45.

1. Introduction

The concept of formal language was first introduced and discussed by linguist, Noam Chomsky in [6]. A formal language is a set of sequence of symbols over some finite alphabet and the formal languages are generated or recognized by abstract devices. Some of the major and useful devices to generate formal languages are formal grammars (regular, context-free and context-sensitive), Petri nets, L-systems and automata [14, 17, 29]. Later various classes of formal languages were introduced, also their relation with its generating devices are initiated and examined. Formal grammars generally called as grammars are precise description of formal languages, that is a grammar is a set of rules for rewriting strings to produce a language and the rewriting begins from a start symbol. The in-depth knowledge of formal languages along with its application in several fields such as pattern matching, syntax, lexical analyzer, picture recognition etc, and its relation with grammars can be found in [14, 28]. Recently, there is an increasing interest in studying and analyzing grammars and Petri nets, which are more applicable and effective devices to generate the formal languages. Petri nets are dynamic graph with two disjoint sets of nodes of which first includes the places represented by circles and the second are transitions represented by rectangles [27]. A Petri net is used to create a simple mathematical framework for modelling concurrent systems and their behaviour. The collection of firing sequences of transitions produced by the Petri net is the primary importance in many applications of Petri nets. In addition to that, they have been used for the description and analysis of system of parallel process [25, 30, 31]. Among the distinct varieties of Petri nets

*Corresponding author
Email addresses: ja8952@srmist.edu.in (A. John Kaspar), sheena.lesley@gmail.com (D. K. Sheena Christy)

doi: 10.22436/jmcs.030.02.01

Received: 2022-04-25 *Revised:* 2022-10-23 *Accepted:* 2022-11-10

structures, one of the Petri net structure called labelled Petri net was introduced in [12]. Their extensions and applications has been studied in many papers [1, 3, 34]. These labelled Petri nets has been useful for investigating the properties and characteristics of formal languages as they can successfully represent and analyse the flow of information and the control of action in concurrent systems. In 1967 [9], Hack formulated the Petri net languages generated by labelled transitions of Petri nets over some alphabet. This was the commencement of a root causing research for the classification of a number of Petri net languages [7, 11, 18, 30]. The hierarchy, complexity and deciedabilty results of Petri net languages are discussed and analyzed in [2, 10]. In [32], another version of labeled Petri net called string token Petri net has been presented and studied by representing the labels of transitions with grammar's rules and tokens by strings over some alphabet. This string token Petri nets led to the study of characterization of family of picture languages recognized by array token Petri nets and its extension to the generation of graphs and trees [15, 16, 22, 23, 33].

Zadeh introduced the concept of fuzzy sets to deal with real life problems having uncertainties, vagueness and imprises data [8, 35]. In [21], fuzzy Petri nets was introduced to reduce the uncertainty problems, which aeries in decision making of industrial models. Later varied extensions of fuzzy Petri nets has been initiated and studied by many researchers [4, 13, 20, 36, 37]. In [19], Zadeh and Lee introduced fuzzy languages generated by fuzzy grammars to reduce the vagueness occuring in formal languages. In addition to that, they extended and discussed the basic principles and results of formal languages to fuzzy languages. Fuzzy languages received a lot of importance since the late seventies and the new class of fuzzy languages with its associated generative fuzzy devices were further introduced and properties were scrutinized by various studies [26]. In [5, 24], The generative tools such as fuzzy L-system, fuzzy grammar and fuzzy Petri nets were discussed and some properties of fuzzy languages has been studied from the aspect of those fuzzy generative tools. The fuzzy Petri net introduced in [5] is a labelled fuzzy Petri net, in which the transitions of Petri nets were labeled by symbols over some alphabet associated with their membership values. The concept of fuzzy languages is used in script recognition, pattern matching, lexical analysis, etc [26].

Motivated by the above works, in this paper, the new class of fuzzy Petri net is introduced and analyzed in detail by labeling the transitions of fuzzy Petri net with production rules of fuzzy grammars. Further, the closure properties of fuzzy Petri net are studied from the perspective of fuzzy grammars with fuzzy languages.

This paper is organized as follows. The basic notions of fuzzy languages, Petri nets and Petri net languages are recalled in Section 2. In Section 3, fuzzy evaluation rules, string token fuzzy Petri net and regular string token fuzzy Petri net are defined. Also, shown the construction of the regular string token fuzzy Petri net from the given fuzzy regular language. The closure properties such as union, concatenation, kleene closure, homomorphism, inverse homomorphism and reversal of the family of fuzzy regular languages generated by regular string token fuzzy Petri net are studied.

2. Preliminaries

In this section, the concept of fuzzy regular grammar and fuzzy regular languages are recalled [19, 26]. Also, the basic definitions of Petri net, labeled Petri net, Petri net language, string token Petri net and fuzzy Petri net are recalled [9, 12, 25, 27, 30, 32]. The notion of regular languages, regular grammar and some useful properties can be found in [14, 28].

Definition 2.1 ([19, 26])**.** A *fuzzy grammar* is a 4-tuple $G = (V_G, T_G, S, P_G)$ where,

1. V_G is a finite collection of non-terminals;
2. T_G is a finite collection of terminals, $(V_G \cap T_G = \emptyset)$;
3. $S \in V_G$ is start symbol; and
4. P_G is a collection of rules called fuzzy production rules, which are of the form $A \xrightarrow{\rho} B$, where $A,\ B \in (V_G \cup T_G)^*$ and $\rho \in (0,\ 1]$ is the grade of membership (membership value) of A given B.

Definition 2.2 ([19, 26])**.** Let $\alpha,\ \beta \in (V_G \cup T_G)^*$. The string β is derivable from α, if there exist $\alpha_1, \dots, \alpha_n \in (V_G \cup T_G)^*$ and $\rho_1, \rho_2, \dots, \rho_n, \rho_{n+1} \in (0, 1]$ such that $\alpha \xrightarrow{\rho_1} \alpha_1$, $\alpha_1 \xrightarrow{\rho_2} \alpha_2, \dots, \alpha_{n-1} \xrightarrow{\rho_n} \alpha_n$, $\alpha_n \xrightarrow{\rho_{n+1}} \beta$ are fuzzy production rules in G.

A derivation chain from α_1 to α_n is an expression $\alpha_1 \overset{\rho_1}{\Longrightarrow} \alpha_2 \overset{\rho_2}{\Longrightarrow} \alpha_3 \dots \alpha_{n-1} \overset{\rho_{n-1}}{\Longrightarrow} \alpha_n$.

Definition 2.3 ([19, 26])**.** A string x of terminals is said to be derivable from a start variable $S \in V_G$, symbolically $S \overset{*}{\Rightarrow} x$, if there is at least one derivation chain from S to x.

The *degree of the string* x is given by $\mu(x) = \sup\{\min(\rho_1, \dots, \rho_{n-1})\}$, where the supremum is taken over all derivation chains from S to x.

The fuzzy language generated by G is denoted by $L(G)$ and it is defined by $L(G) = \{(x, \mu(x))/S \overset{*}{\Rightarrow} x; S \in V_G \text{ and } x \in V_G^*\}$.

Definition 2.4 ([19, 26])**.** A fuzzy grammar $G = (V_G, T_G, S, P_G)$ is said to be *fuzzy regular grammar*, if all the fuzzy production rules in G are of the form $A \xrightarrow{\rho} yB$ or $A \xrightarrow{\rho} y$, where $A, B \in V_G, y \in T_G^*$ and $\rho \in (0, 1]$.

Definition 2.5 ([19, 26])**.** The collection of all strings $y \in T_G^*$ generated by the fuzzy regular grammar G is called the *fuzzy regular language* and it is denoted as *L(G).*

Definition 2.6 ([15, 30])**.** A Petri net is a quadruple $C = (P, T, I, O)$, where

- $P = \{p_i : i = 1, 2, \dots, n\}$ is a finite collection of *places;*
- $T = \{t_j : j = 1, 2, \dots, m\}$ is a finite collection of *transitions*, where P and T are disjoint and $P \cup T \neq \emptyset$;
- *input function* I, that maps transitions to a collection of places (i.e., $I : T \to 2^P$);
- *output function* O, that maps a collection of places to transitions (i.e., $O : T \to 2^P$).

Definition 2.7 ([15, 30])**.** A generalized Petri net is a quintuple $N = (P, T, I, O, M_0)$ where,

- P, T, I, and O are same as defined in Definition 2.6;
- M_0 is a marking called the *initial marking*, in which a marking is a mapping from places P to $\{0, 1, 2, \dots\}$.

Definition 2.8 ([12, 15, 30])**.** A labeled Petri net is a hextuple $N = (P, T, I, O, M_0, \gamma)$, where P, T, I, O, M_0 are same as defined in Definition 2.7 and γ is a labeling function, which assigns labels to the transitions.

Definition 2.9 ([22, 23])**.** A String Token Petri Net (STPN) is a hextuple $N = (P, T, V, A, R(t), M_0)$, where

- $P = \{p_i : i = 1, 2, \dots, n\}$ is a finite collection of *places;*
- $T = \{t_j : j = 1, 2, \dots, m\}$ is a finite collection of *transitions* and each t_j is the label of evolution rules;
- V is a finite non-empty collection of *alphabets* consisting of terminals (denoted in lower case letters) and non-terminals (denoted in upper case letters);
- $A \subseteq (T \times P) \cup (P \times T)$ is a collection of *arcs* (flow relation);
- $R(t)$ is the collection of evolution rules used to label each transition $t_j; j = 1, 2, \dots, m$ of T;
- $M_0 : P \to$ (a string over V) is an *initial marking;*

where, P and T are disjoint and $P \cup T \neq \emptyset$. The collection of all strings generated by the STPN N is called the language generated by the STPN.

Definition 2.10 ([24])**.** A *fuzzy language* generating Petri net is a sep-tuple $(C, \gamma, \Sigma, M_0, M_F, \omega, \oplus)$, where

- C, γ, and M_0 are same as defined in Definition 2.8;
- Σ is a finite collection *alphabets*, used to label the transitions;

- M_F is a collection of *final markings;*
- $\omega : T \to [0,1]$ assigns membership values to every transitions $t_j; j = 1, 2, \dots, m$ of T; and
- $\oplus$ is a t-norm obtains the membership value (grade of membership) of the generated string based on the values of the switched transitions, which is the grade of membership of the generated string to the fuzzy language.

3. Regular string token fuzzy Petri net

In this section, fuzzy evolution rules, string token fuzzy Petri net and regular string token fuzzy Petri net have been introduced and described with example. Furthermore, it has been proved that for every fuzzy regular language there exists a regular string token fuzzy Petri net.

Definition 3.1. *Fuzzy Evolution Rules* (FER) for String Token Fuzzy Petri Net (STFPN) are as follows.

- *Reflexive rule*: $a \xrightarrow{1} a$, i.e., there is no change in the string replacement.
- *Insertion rule*: $\lambda \xrightarrow{1} a$, an empty string ($\lambda$) is replaced by a string over V. It can be done in either leftmost or rightmost side of the string.
- *Deletion rule*: $a \xrightarrow{0} \lambda$, a string over V is replaced by an empty string. It can be done either in leftmost or rightmost side of the string.
- *Substitution rule*: $a \xrightarrow{\rho} b, \rho \in [0,1]$, a string over V is replaced by another string over V,

where $a \in V$, $b \in V^*$ and λ is the empty string.

Definition 3.2. A *String Token Fuzzy Petri Net* (STFPN) is a 8-tuple $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where

- $P = \{p_i : i = 1, 2, \dots, n\}$ is a finite collection of *places;*
- $T = \{t_j : j = 1, 2, \dots, m\}$ is a finite collection of *transitions*, where each t_j is the label of fuzzy evolution rules;
- V is a finite non-empty collection consists of terminals (denoted as V_{tr}, where the elements are represented by lower case letters) and non-terminals (denoted as V_{ntr}, where the elements are represented by upper case letters);
- $A \subseteq (T \times P) \cup (P \times T)$ is a collection of *arcs* (flow relation) and an arc from any p (place) to t (transition) is denoted by $p \mapsto t$ (similarly, $t \mapsto p$);
- $F \subseteq P$ is a collection of *final places*, which contains only strings of terminals;
- $\mathfrak{R}(t)$ is the collection of fuzzy evolution rules (FRE) labeled with each transition $t_j; j = 1, 2, \dots, m$ of T;
- $M_0 : P \to$ (a string over V) is the *initial marking;*
- $M_F : P \to$ (a string of terminals over V) is the set of *final markings* (also called as the reachable markings).

where, $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$.

Definition 3.3. A state or marking in an STFPN is altered according to the following transition (firing) rule to replicate the dynamic system's behavior.

(i) A transition (say t) is termed an enabled transition, if the tokens of every input place (say p) of a transition t possesses a string appearing on the left-side expression of the FER labeled with t (for example, the transition t labelled by FRE $AxB \xrightarrow{\rho} BCy$ is enabled. Then the string token of its input place p of t is a string AxB).

(ii) A transition t can be fired, if it is enabled (depending on whether the event actually takes place).

(iii) The string token (say X) is removed from all input place (say p) of a transition (say t) and altered as the string token Y in the output place of t, if the enabled transition t is fired (for example: if $t : AxB \xrightarrow{\rho} BCy$ is an enabled transition and a string $wAxBz$ on the input place (say p_1) of t is altered as the string $wBCyz$ on the output place (say p_2) of t, after t fires).

If a transition t in T is labelled with a FRE $A \xrightarrow{\rho} B$ then firing of the transition t from an input place to a output place is denoted by $A \mathrel{\vert\!\frac{\rho}{t}} B$, where $A, B \in V^*$; $\rho \in [0, 1]$. A firing sequence from X_1 to X_n with transitions $t_1, t_2, \dots, t_{n-1} \in T$ is denoted by $X_1 \mathrel{\vert\!\frac{\rho_1}{t_1}} X_2 \mathrel{\vert\!\frac{\rho_2}{t_2}} X_3 \mathrel{\vert\!\frac{\rho_3}{t_3}} \cdots X_{n-1} \mathrel{\vert\!\frac{\rho_{n-1}}{t_{n-1}}} X_n$, where $X_i \in V^*; i = 1, \dots, n$ and $\rho_i \in [0, 1]; i = 1, \dots, n-1$. The degree of derivability (membership value) of X_1 to X_n is obtained by $d(X_1 \overset{\star}{\vdash} X_n) = \sup \min\{\rho_1, \rho_2, \dots, \rho_{n-1}\}$, where the supremum is taken over all the firing sequence from X_1 to X_n.

Note that, $\overset{\star}{\vdash}$ is the transitive closure of $\vdash$.

Definition 3.4. A string w of terminals is generated by the STFPN $\mathcal{N}^f$, if there exists atleast one firing sequence that reaches the final marking for the string w from the initial marking M_0 with $d(M_0 \overset{\star}{\vdash} w) > 0$. The membership value of the string is denoted by $d(w)$ (also, $d_{\mathcal{N}^f}(w)$) and is obtained by $d(w) = d(M_0 \vdash w)$.

The fuzzy language generated by the STFPN $\mathcal{N}^f$ is the collection of all strings of terminals generated by the STFPN $\mathcal{N}^f$ and it is denoted by $L(\mathcal{N}^f)$. That is,

$$L(\mathcal{N}^f) = \{w/w \in V_{tr}^* \text{ and } d(M_0 \overset{\star}{\vdash} w) > 0\}.$$

Definition 3.5. A *Regular STFPN* (RSTFPN) is also a 8-tuple $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, in which P, T, V, A, F, M_0, M_F are all same as defined in Definition 3.2 and identity, insertion and deletion rules of FER $\mathfrak{R}(t)$ are same as defined in Definition 3.1 in which the substitution rules are of the form $\alpha \xrightarrow{\rho} y\beta$ or $\alpha \xrightarrow{\rho} y$, where $\alpha, \beta \in V_{ntr}, y \in V_{tr}^*$ and $\rho \in (0, 1]$.

Example 3.6. Consider a fuzzy regular language $L(\mathcal{N}^f) = \{(w, 0.5)/w = a^n b^m; n \geqslant 2, m \geqslant 3\}$ generated by the RSTFPN $N^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, in which, $P = \{p_1, p_2, p_3, p_4, p_5\}$, $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $V = \{S, A, B, C, a, b\}$, $A = \{p_1 \mapsto t_1, t_1 \mapsto p_2, p_2 \mapsto t_2, t_2 \mapsto p_2, p_2 \mapsto t_3, t_3 \mapsto p_3, p_3 \mapsto t_4, t_4 \mapsto p_4, p_4 \mapsto t_5, t_5 \mapsto p_4, p_4 \mapsto t_6, t_6 \mapsto p_5\}$, $F = \{p_5\}$, M_0: initial marking, M_F: reachable marking, and $\mathfrak{R}(t) = \{S \xrightarrow{0.8} aaA, A \xrightarrow{0.7} aA, A \xrightarrow{0.9} B, B \xrightarrow{0.8} bbC, C \xrightarrow{0.6} aC, C \xrightarrow{0.6} a\}$. The RSTFPN $\mathcal{N}^f$ generating the fuzzy regular language $L(\mathcal{N}^f)$ is illustrated in Figure 1.

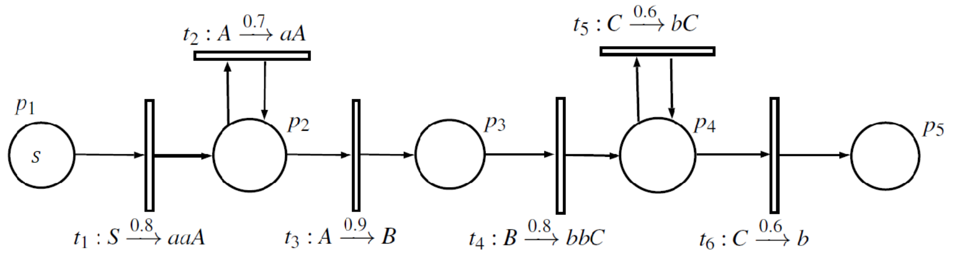

$t_2 : A \xrightarrow{0.7} aA$
$t_5 : C \xrightarrow{0.6} bC$
p_1
S
p_2
p_3
p_4
p_5
$t_1 : S \xrightarrow{0.8} aaA$
$t_3 : A \xrightarrow{0.9} B$
$t_4 : B \xrightarrow{0.8} bbC$
$t_6 : C \xrightarrow{0.6} b$

Figure 1: RSTFPN $\mathcal{N}^f$.

Let $y = aabbb$ in $L(\mathcal{N}^f)$, the firing sequence of the string y and the membership value of the string is given below:

$$S \left|\frac{0.8}{t_1}\right. aaA \left|\frac{0.9}{t_3}\right. aaB \left|\frac{0.8}{t_4}\right. aabbC \left|\frac{0.6}{t_6}\right. aabbb, \quad d(y) = \max\min(0.8, 0.9, 0.8, 0.6) = 0.6 > 0.$$

Therefore the string $y = aabbb$ generated by $\mathcal{N}^f$ with membership value 0.6 is in $L(\mathcal{N}^f)$.

Definition 3.7. A fuzzy language generated by the RSTFPN $\mathcal{N}^f$ is called the fuzzy regular language generated by $\mathcal{N}^f$ and is denoted by $L(\mathcal{N}^f)$.

Theorem 3.8. *For every fuzzy regular language* L, *there exists a RSTFPN* $\mathcal{N}^f$ *such that* $L = L(\mathcal{N}^f)$.

Proof. Let L be a given fuzzy regular language generated by the fuzzy regular grammar $G = (V_G, T_G, S, P_G)$, where V_G, T_G, S are same as defined in Definition 3.5 and the fuzzy production rules in P_G are of the form $S \xrightarrow{\rho} zB$, $B \xrightarrow{\gamma} yB$, $S \xrightarrow{\beta} yC$, $B \xrightarrow{\alpha} yD$ $D \xrightarrow{\omega} w$ and $C \xrightarrow{\eta} y$ where, $y, z \in T_G^*$; $S, B, C, D \in V_G$ and $\rho, \gamma, \beta, \alpha, \omega, \eta \in [0, 1]$.

The fuzzy production rules P_G of G is partitioned as follows, which has been used in the construction of RSTFPN $\mathcal{N}^f = (P', T', V', A', F', \mathfrak{R}'(t), M'_0, M'_F)$.

i) Collection of all fuzzy production rules are of the form $S \xrightarrow{\rho} zB$, $B \xrightarrow{\gamma} yB$, $S \xrightarrow{\beta} yC$ and $B \xrightarrow{\alpha} yD$ are called as NT-rules, i.e., the collection of non-terminal rules.

ii) Collection of all other fuzzy production rules are called as T-rules, i.e., the collection of terminal rules.

iii) Collection of all fuzzy production rules whose leftside has start variable (say S) of G are called as S-rules.

Among the NT-rules, collection of all fuzzy production rules are of the form $B \xrightarrow{\beta} yB$ (i.e., loop rules) are called as LNT-rules and collection of all other fuzzy production rules are called as WLNT-rules.

Among the S-rules, collection of all NT-rules are called as SNT-rules and all other rules are called as ST-rules.

Step 1: If S is the start symbol of G then construct a place p_S with S as a token.

Step 2: Let there are n rules in the set of S-rules and let t_i be the tag of the n rules in the set of S-rules, where $i = 1, 2, \ldots, n$.

```
for ($i = 1,\ i \leq n,\ i = i + +$) do
    if $t_i \in SLNT$ then
        transition label = $t_{SLNT\ i}$;
        input place = $P_S$;
        output place = $P_S$;
        firing times = any;
    else
        if $t_i \in SWLNT$ then
            transition label = $t_{SWLNT\ i}$;
            input place = $P_S$;
            output place = $P_{SW\ i}$;
            firing times = one;
        else
            if $t_i \in ST$ then
                transition label = $t_{ST\ i}$;
                input place = $P_S$;
                output place = $P_{ST\ i}$;
                firing times = one;
            end
        end
    end
end
```

All the strings in $P_{Sw_i}, i = 1, 2, \ldots, n$ has at least one non-terminal in it. Let C_j be the leftmost non-terminal in Sw_i. Now, collect all C_j-rules partition them as C_jLNT, C_jWLNT and C_jT. Let C_j has n number of rules.

Step 3: The Regular STFPN $\mathcal{N}^f = (P', T', V', A', F', \mathfrak{R}'(t), M'_0, M'_F)$, in which P' is the collection all constructed places, T' the collection of all labeled transitions used so far, $V' = V_G \cup T_G$, A' the collection all arcs, $\mathfrak{R}'(t) = P_G$, M'_0 is the initial marking and M'_F is the collection of reachable markings.

```
while $i \le n$ do
    Search($P_{Sw_i}, C_j$);
    for ($j = 1,\ j \le n,\ j = j + +$) do
        if $t_j \in C_jLNT$ then
            transition label = $t_{C_jLNT\ j}$;
            input place = $P_{Sw_j}$;
            output place = $P_{Sw_j}$;
            firing times = any;
        else
            if $t_j \in C_jWLNT$ then
                transition label = $t_{C_jWLNT\ j}$;
                input place = $P_{PSw_j}$;
                output place = $P_{C_jW\ j}$;
                firing times = one;
            else
                if $t_j \in C_jT$ then
                    transition label = $t_{C_jT\ j}$;
                    input place = $P_{Sw_j}$;
                    output place = $P_{C_jT\ j}$;
                    firing times = one;
    $i = i + 1$;
```

Now, collection of all places $P_{ST_i}, P_{C_jT_j}, \ldots$ are called as final places F', since it leads to the terminal strings and collection of all the places so far are called as P'.

Thus, for every fuzzy regular language L there exists a Regular String Token Fuzzy Petri Net $\mathcal{N}^f$ such that $L = L(\mathcal{N}^f)$. □

Example 3.9. Consider the fuzzy regular language $L_1 = \{a^n b / n \geqslant 0; a, b \in \Sigma\}$ with membership value 0.6 when $n = 1$ and 0.4 when $n \geqslant 2$ generated by the fuzzy regular grammar $G = (\{S, S_1\}, \{a, b\}, S, P_G)$, where $P_G = \{S \xrightarrow{0.7} aS_1, S_1 \xrightarrow{0.4} aS_1, S \xrightarrow{0.6} b\}$.

Among the rules in P_G, $S \xrightarrow{0.7} aS_1, S_1 \xrightarrow{0.4} aS_1$ are NT-rules and $S_1 \xrightarrow{0.6} b$ is a T-rule. Also, among the NT-rules $S_1 \xrightarrow{0.7} aS_1$ is a LNT-rule and $S \xrightarrow{0.4} aS_1$ is a WLNT-rule. Here, $S \xrightarrow{0.7} aS_1$ is the SWLNT-rule, since S is the start symbol of G and also a NT-rule.

By step 1 of Theorem 3.8, construct a place p_S with S as a string token in it (Figure 2).

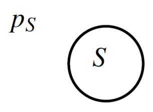


Figure 2:

By step 2 of Theorem 3.8, label the rule $S \xrightarrow{0.7} aS_1$ as t_{SWLNT1}, since it is a SWLNT-rule. Now, construct a transition $t_{SWLNT1} : S \xrightarrow{0.7} aS_1$ with input place and output place as p_S and p_{aS_1} (Figure 3), respectively.

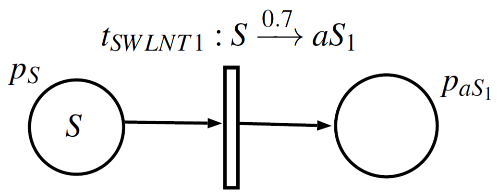
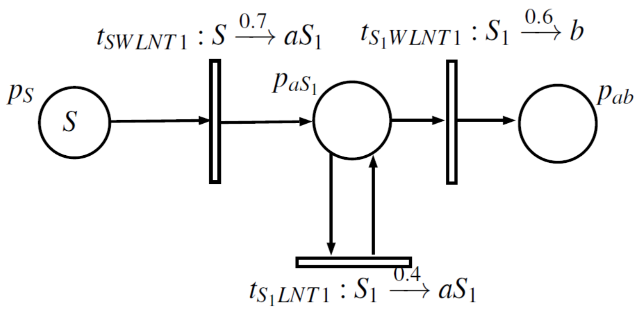


Figure 3:

By step 3 of Theorem 3.8, collect all the S_1-rules, since S_1 is the leftmost non-terminal in aS_1. Label the the rule $S_1 \xrightarrow{0.4} aS_1$ as t_{S_1LNT1}, since it is a S_1LNT-rule. Now, construct a transition $t_{S_1LNT1} : S_1 \xrightarrow{0.4} aS_1$ with input place and output place as p_{aS_1}. Also, label the rule $S_1 \xrightarrow{0.6} b$ as t_{S_1T1}, since it is a S_1T1-rule. Now, construct a transition $t_{S_1T1} : S_1 \xrightarrow{0.6} b$ with input place and output place as p_{aS_1} and p_{ab}, respectively. The complete RSTFPN $\mathcal{N}^f$ is illustrated in Figure 4.



$t_{SWLNT1} : S \xrightarrow{0.7} aS_1$
$t_{S_1WLNT1} : S_1 \xrightarrow{0.6} b$
p_S
S
p_{aS_1}
p_{ab}
$t_{S_1LNT1} : S_1 \xrightarrow{0.4} aS_1$

Figure 4: RSTFPN $\mathcal{N}^f$.

Thus, the RSTFPN $\mathcal{N}^f$ that generates the given fuzzy regular language L is $\mathcal{N}^f = (P, V, T, A, F, \mathfrak{R}(t), M_0, M_F)$, where,

$$\begin{aligned}
P &= \{p_S, p_{aS_1}, p_{ab}\},\\
V &= \{S, S_1, a, b\},\\
T &= \{t_{SWLNT1}, t_{S_1LNT1}, t_{S_1T1}\},\\
A &= \{p_S \mapsto t_{SWLNT\,1}, t_{SWLNT\,1} \mapsto p_{aS_1}, p_{aS_1} \mapsto t_{S_1LNT\,1},\\
&\qquad t_{S_1LNT\,1} \mapsto p_{aS_1}, p_{aS_1} \mapsto t_{S_1WLNT\,1}, t_{S_1WLNT\,1} \mapsto p_{ab}\},\\
F &= \{p_{ab}\},\\
\mathfrak{R}(t) &= \{t_{SWLNT1} : S \xrightarrow{0.7} aS_1, t_{S_1LNT1} : \xrightarrow{0.4} aS_1, t_{S_1T1} : S_1 \xrightarrow{0.6} b\},\\
M_0 &= (S, \varepsilon, \varepsilon) \text{ and } M_F = \{\varepsilon, \varepsilon, p_{ab}\}.
\end{aligned}$$

It is easy to see that $L(G) = L(\mathcal{N}^f)$.

4. Closure properties

In this section, the closure properties such as, union, concatenation, kleene closure, reversal, homomorphism and inverse homomorphism are defined and proved that the fuzzy regular languages generated by the RSTFPN are closed under union, concatenation, kleene closure, reversal, homomorphism and inverse homomorphism.

Definition 4.1. Let $L(\mathcal{N}^f)$, $L(\mathcal{N}^f_1)$ and $L(\mathcal{N}^f_2)$ be the fuzzy regular languages generated by the RSTFPN $\mathcal{N}^f$, $\mathcal{N}^f_1$ and $\mathcal{N}^f_2$, respectively then the union of two fuzzy regular languages $L(\mathcal{N}^f_1)$ and $L(\mathcal{N}^f_2)$ is also a fuzzy regular language defined by

$$d_{L(\mathcal{N}_1^f)\cup L(\mathcal{N}_2^f)}(w) = \max(d_{L(\mathcal{N}_1^f)}(w),\ d_{L(\mathcal{N}_2^f)}(w)), \text{ where w is a string of terminals}$$

and is denoted by $L(\mathcal{N}_1^f) \cup L(\mathcal{N}_2^f)$.

The intersection of $L(\mathcal{N}_1^f)$ and $L(\mathcal{N}_2^f)$ is also a fuzzy regular language defined by

$$d_{L(\mathcal{N}_1^f)\cap L(\mathcal{N}_2^f)}(w) = \min(d_{L(\mathcal{N}_1^f)}(w),\ d_{L(\mathcal{N}_2^f)}(w)), \text{ where w is a string of terminals}$$

and is denoted by $L(\mathcal{N}_1^f) \cap L(\mathcal{N}_2^f)$.

The concatenation of $L(\mathcal{N}_1^f)$ and $L(\mathcal{N}_2^f)$ is also a fuzzy regular language defined by

$$d_{L(\mathcal{N}_1^f)L(\mathcal{N}_2^f)}(w) = \sup_{w_1} \min(d_{L(\mathcal{N}_1^f)}(w),\ d_{L(\mathcal{N}_2^f)}(w)),$$

where w is a string of terminals the supremium has taken over all the terminal strings of $L(\mathcal{N}_1^f)$ and is denoted by $L(\mathcal{N}_1^f)L(\mathcal{N}_2^f)$.

The kleene closure of $L(\mathcal{N}_1^f)$ is also a fuzzy regular language denoted by $L(\mathcal{N}_1^{f*})$ and is defined by

$$L(\mathcal{N}_1^{f*}) = L(\mathcal{N}_1^{f*})^1 L(\mathcal{N}_1^{f*})^2 \cdots L(\mathcal{N}_1^{f*})^i \cdots .$$

Theorem 4.2. *The fuzzy regular languages generated by regular string token fuzzy Petri nets are closed under union.*

Proof. Let L_1 and L_2 be two fuzzy regular languages generated by regular string token fuzzy Petri net $\mathcal{N}_1^f = (P_1, T_1, V_1, A_1, F_1 \mathfrak{R}_1(t), M_{0_1}, M_{F_1})$ such that $L_1 = L(\mathcal{N}_1^f)$ and a regular string token fuzzy Petri net $\mathcal{N}_2^f = (P_2, T_2, V_2, A_2, F_2, \mathfrak{R}_2(t), M_{0_2}, M_{F_2})$ such that $L_2 = L(\mathcal{N}_2^f)$.

The construction of the RSTFPN $\mathcal{N}^f$ such that $L(\mathcal{N}^f) = L(\mathcal{N}_1^f) \cup L(\mathcal{N}_2^f)$ is done by the following three steps. First, the string tokens S_1 and S_2 are removed from the start places say p_{S_1} and p_{S_2} of the nets $\mathcal{N}_1^f$ and $\mathcal{N}_2^f$, respectively. In the second step, a new place p_S is constructed with string token S in it. Finally, add the transition labeled, t_α with insertion rule $S \xrightarrow{1} S_1$, whose input and output place as p_S and p_{S_1}, respectively. Also, add another transition labeled, t_β with insertion rule $S \xrightarrow{1} S_2$, whose input and output place as p_S and p_{S_2}, respectively. Once the process is initiated, either $\mathcal{N}_1^f$ or $\mathcal{N}_2^f$ continues to operate usually, after firing of the transitions t_α and t_β. The membership value of the string is obtained by taking maximum of its membership value in $L(\mathcal{N}_1^f)$ and $L(\mathcal{N}_2^f)$.

Formally it is defined as follows: construct a regular string token fuzzy Petri net $\mathcal{N}^f$ that generates the fuzzy regular language $L_1 \cup L_2$ such that $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where

$$\begin{aligned} &P = P_1 \cup P_2 \cup \{p\}, && T_1 \cup T_2 \cup \{t_\alpha, t_\beta\}, \\ &V = V_1 \cup V_2 \cup \{S\} \setminus \{S_1, S_2\}, && A = A_1 \cup A_2 \cup \{p \mapsto t_\alpha, p \mapsto t_\beta, t_\alpha \mapsto p_{S_1}, t_\beta \mapsto p_{S_2}\}, \\ &F = F_1 \cup F_2, && \mathfrak{R}(t) = \mathfrak{R}_1(t) \cup \mathfrak{R}_2(t) \cup \{t_\alpha : S \xrightarrow{1} S_1, t_\beta : S \xrightarrow{1} S_2\}, \end{aligned}$$

and M_0 is the initial marking and $M_F = \{M_{F_1} \cup M_{F_2}\}$.

The same procedure is extended to $L_1, L_2, \dots, L_n$, i.e., let $L_1, L_2, \dots, L_n$ be fuzzy regular languages generated by regular string token fuzzy Petri nets, then $L = \bigcup_{i=1}^{n} L_i$ is also a fuzzy regular languages generated by regular string token fuzzy Petri net. □

Theorem 4.3. *The fuzzy regular languages generated by regular string token fuzzy Petri nets are closed under concatenation.*

Proof. Let L_1 and L_2 be the two fuzzy regular languages generated by regular string token fuzzy Petri net $\mathcal{N}_1^f = (P_1, T_1, V_1, A_1, F_1, \mathfrak{R}_1(t), M_{0_1}, M_{F_1})$ such that $L_1 = L(\mathcal{N}_1^f)$ and regular string token fuzzy Petri net $\mathcal{N}_2^f = (P_2, T_2, V_2, A_2, F_2, \mathfrak{R}_2(t), M_{0_2}, M_{F_2})$ such that $L_2 = L(\mathcal{N}_2^f)$.

The construction of the RSTFPN $\mathcal{N}^f$ such that $L(\mathcal{N}^f) = L(\mathcal{N}_1^f)L(\mathcal{N}_2^f)$ is done by the following two steps. First, a new place p is constructed at the end of the net $\mathcal{N}_1^f$ with strings generated from the net $\mathcal{N}_1^f$ as token in it. In the second step, add a transition labeled, t_α with insertion rule $\lambda \xrightarrow{1} S_2$, whose input and output place as p and p_{S_2} (start place of $\mathcal{N}_2^f$), respectively. Any string in $L(\mathcal{N}^f)$ is first operated by the net $\mathcal{N}_1^f$ followed by the net $\mathcal{N}_2^f$. The membership value of the string is obtained by taking maximum of its membership value in $L(\mathcal{N}_1^f)$ and $L(\mathcal{N}_2^f)$.

Formally it can be written as, construct a regular string token fuzzy Petri net $\mathcal{N}^f$ that generates the fuzzy regular language L_1L_2 such that $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where

$$\begin{aligned} &P = P_1 \cup P_2\{p\text{: a place containing the words of } L_1\}, && T_1 \cup T_2 \cup \{t_\alpha\},\\ &V = V_1 \cup V_2 \cup \{\lambda\}, && A = A_1 \cup A_2 \cup \{p \mapsto t_\alpha, t_\alpha \mapsto p_{S_2}\},\\ &F = F_2, && \mathfrak{R}(t) = \mathfrak{R}_1(t) \cup \mathfrak{R}_2(t) \cup \{t_\alpha : \lambda \xrightarrow{1} S_2\}, \end{aligned}$$

and M_0 is the initial marking and $M_F = \{M_{F_2}\}$.

The same procedure is extended to $L_1, L_2, \dots, L_n$, i.e., Let $L_1, L_2, \dots, L_n$ be fuzzy regular languages generated by regular string token fuzzy Petri nets then $L = L_1L_2 \dots L_n$ is also a fuzzy regular languages generated by regular string token fuzzy Petri net. □

Theorem 4.4. *The fuzzy regular languages generated by regular string token fuzzy Petri nets are closed under kleene closure.*

Proof. Let L_1 be a fuzzy regular language generated by a regular string token fuzzy Petri net $\mathcal{N}_1^f = (P_1, T_1, V_1, A_1, F_1, \mathfrak{R}_1(t), M_{0_1}, M_{F_1})$ such that $L_1 = L(\mathcal{N}_1^f)$.

The construction and process of the net $L(\mathcal{N}^f)$ is same as stated in Theorem 4.3, whereas the transition labeled t_α has insertion rule $\lambda \xrightarrow{1} S_1$ with input and output place as p and p_{S_1} (start place of $\mathcal{N}_1^f$), respectively.

The formal way is as follows: construct a regular string token fuzzy Petri net $\mathcal{N}^f$ that generates the fuzzy regular language L_1^* such that $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where

$$\begin{aligned} &P = P_1 \cup \{p\text{: a place containing the words of } L_1\}, && T = T_1 \cup \{t_\alpha\}, && V = V_1 \cup \{\varepsilon\},\\ &A = A_1 \cup \{p \mapsto t_\alpha, t_\alpha \mapsto p_{S_1}\}, && F = F_1, && \mathfrak{R}(t) = \mathfrak{R}_1(t) \cup \{t_\alpha : \lambda \xrightarrow{1} S_1\}, \end{aligned}$$

and M_0 is the initial marking and $M_F = \{M_{F_1}\}$. □

Theorem 4.5. *The fuzzy regular languages generated by regular string token fuzzy Petri nets are closed under reversal.*

Proof. Let L_1 be a fuzzy regular language then there exists a regular string token fuzzy Petri net $\mathcal{N}_1^f = (P_1, T_1, V_1, A_1, F_1, \mathfrak{R}_1(t), M_{0_1}, M_{F_1})$ such that $L_1 = L(\mathcal{N}_1^f)$.

Now, construct a regular string token fuzzy Petri net $\mathcal{N}^f$ that generates the fuzzy regular language L_1^R such that $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where $P = P_1$, $T = T_1$, $V = V_1$, $F = F_1$, $A = A_1$,

$$\mathfrak{R}(t) = \begin{cases} A \xrightarrow{\rho} By^R, & \text{if } A \xrightarrow{\rho} yB; A, B \in V_{ntr}, y \in V_{tr}^*,\\ A \xrightarrow{\rho} y^R, & \text{if } A \xrightarrow{\rho} y; A \in V_{ntr}, y \in V_{tr}^*, \end{cases}$$

and M_0 is the initial marking and $M_F = \{M_{F_1}\}$. The concept is that, first the terminal strings occurs in the leftside of the rules are written in the reverse order and then they are labeled by the transitions t_j from T of the net $\mathcal{N}^f$ with same the membership value. □

Definition 4.6. Let V^* and V_1^* be two given alphabets. A mapping $\hbar : V^* \to V_1^*$ is called a *homomorphism*, if it satisfies $\hbar(yz) = \hbar(y)\hbar(z)$ for all $y, z \in V_{tr}^*$. The homomorphism of the given language L is denoted by $\hbar(L)$ obtained from the set $\hbar(L) = \{\hbar(x)/x \in L\}$.

Theorem 4.7. *The fuzzy regular languages generated by regular string token fuzzy Petri nets are closed under homomorphism and inverse homomorphism.*

Proof. Let L_1 be a fuzzy regular language generated by a regular string token fuzzy Petri net $\mathcal{N}_1^f = (P_1, T_1, V_1, A_1, F_1, \mathfrak{R}_1(t), M_{0_1}, M_{F_1})$ such that $L_1 = L(\mathcal{N}_1^f)$.
(i) Construct a regular string token fuzzy Petri net $\mathcal{N}^f$ that generates the fuzzy regular language $\hbar(L_1)$ such that $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where $P = P_1$, $T = T_1$, $V = V_1$, $A = A_1$, $F = F_1$,

$$\mathfrak{R}(t) = \begin{cases} A \xrightarrow{\rho} \hbar(y)B, & \text{if } A \xrightarrow{\rho} yB; A, B \in V_{ntr}, y \in V_{tr}^*, \\ A \xrightarrow{\rho} \hbar(y), & \text{if } A \xrightarrow{\rho} y; A \in V_{ntr}, y \in V_{tr}^*, \end{cases}$$

and M_0 is the initial marking and $M_F = \{M_{F_1}\}$.
(ii) Construct a regular string token fuzzy Petri net $\mathcal{N}^f$ that generates the fuzzy regular language $\hbar^{-1}(L_1)$ such that $\mathcal{N}^f = (P, T, V, A, F, \mathfrak{R}(t), M_0, M_F)$, where $P = P_1$, $T = T_1$, $V = V_1$, $A = A_1$, $F = F_1$,

$$\mathfrak{R}(t) = \begin{cases} A \xrightarrow{\rho} \hbar^{-1}(y)B, & \text{if } A \xrightarrow{\rho} yB; A, B \in V_{ntr}, y \in V_{tr}^*, \\ A \xrightarrow{\rho} \hbar^{-1}(y), & \text{if } A \xrightarrow{\rho} y; A \in V_{ntr}, y \in V_{tr}^*, \end{cases}$$

and M_0 is the initial marking and $M_F = \{M_{F_1}\}$.

The concept is that, first the homomorphism (inverse homomorphism) is applied to the terminal strings occurs in the left-side of the rules and then they are labeled by the transitions t_j from T of the net $\mathcal{N}^f$ with same the membership value. □

5. Conclusion

In this paper, a new model, regular string token fuzzy Petri nets to generate fuzzy regular languages has been introduced and discussed. Also, given the construction of regular string token fuzzy Petri net from given fuzzy regular language and some closure properties such as union, catenation, kleene closure, homomorphism, inverse homomorphism and reversal of the languages generated by regular string token fuzzy Petri nets has been established. The scope of this study is to extend the concept to fuzzy context-free languages and fuzzy context-sensitive languages. Also, the study can be further taken up to the complexity theory of array token fuzzy Petri nets. Moreover, find the possible applications of the concept in approximate pattern matching and image processing.

References

[1] A. Al-Ajeli, D. Parker, *Fault diagnosis in labelled Petri nets: a Fourier-Motzkin based approach*, Automatica J. IFAC, **132** (2021), 7 pages. 1

[2] P. T. An, *A complexity characteristic of Petri Net languages*, Acta Math. Vietnam., **24** (1999), 157–167. 1

[3] P. T. An, P. V. Thao, *On capacity of labeled Petri Net languages*, Vietnam J. Math., **27** (1999), 231–240. 1

[4] J. Cardoso, C. Heloisa, *Fuzziness in Petri nets*, Springer Science & Business Media, (1998). 1

[5] S. R. Chaudhari, D. D. Komejwar, *On fuzzy regular grammars*, Adv. Fuzzy Syst., **6** (2011), 89–103. 1

[6] N. Chomsky, *Three models for the description of language*, IRE Trans. Inform. Theory, **2** (1956), 113–124. 1

[7] J. Dassow, G. Mavlankulov, M. Othman, S. Turaev, M. H. Selamat, R. Stiebe, *Grammars controlled by Petri Nets*, In: Petri Nets Manufacturing and Computer Science, InTech, (2012), 337–358. 1

[8] G. J. Klir, B. Yuan, *Fuzzy sets and fuzzy logic*, New Jersey: Prentice hall, (1995). 1

[9] M. Hack, *Petri Net languages*, Technical report, Massachusetts Institute of Technology, (1976). 1, 2

[10] M. Jantzen, *On the hierarchy of Petri Net languages*, RAIRO Inform. Théor., **13** (1979), 19–30. 1

[11] M. Jantzen, M. Kudlek, G. Zetzsche, *Language classes defined by concurrent finite automata*, Fund. Inform., **85** (2008), 267–280. 1

[12] M. Jantzen, G. Zetzsche, *Labeled Step Sequences in Petri Nets*, In: Applications and Theory of Petri Nets, Springer, Berlin, (2008), 270–287. 1, 2, 2.8

[13] W. Jiang, K.-Q. Zhou, A. Sarkheyli-Hägele, A. M. Zain, *Modeling, reasoning, and application of fuzzy Petri net model: a survey*, Artif. Intell. Rev., **55** (2022), 1–39. 1

[14] J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Pearson Education Ltd., 3rd edition, (2014). 1, 2

[15] T. Kalyani, T. T. Raman, D. G. Thomas, K. Bhuvaneswari, P. Ravichandran, *Triangular array token Petri Net and P system*, In International Conference on Membrane Computing, Springer, Cham, (2020), 78–93. 1, 2.6, 2.7, 2.8

[16] T. Kamaraj, D. Lalitha, D. G. Thomas,*A Study on Expressiveness of a Class of Array Token Petri Nets*, Proceedings of the Third International Conference on Soft Computing for Problem Solving, Springer, New Delhi, (2014), 457–469. 1

[17] A. Lindenmayer, *Developmental Systems without Cellular Interactions, their languages and Grammars*, J. Theoret. Biol., **30** (1971), 455—484. 1

[18] S. Lafortune, H. Yoo, *Some results on Petri Net languages*, IEEE Trans. Automat. Control, **35** (1990), 482–485. 1

[19] E. T. Lee, L. A. Zadeh, *Note on fuzzy languages*, Inf. Sci., **1** (1969), 421–434. 1, 2, 2.1, 2.2, 2.3, 2.4, 2.5

[20] H. Li, J.-X. You, H.-C. Liu, G. Tian, *Acquiring and sharing tacit knowledge based on interval 2-tuple linguistic assessments and extended fuzzy Petri nets*, Internat. J. Uncertain. Fuzziness Knowledge-Based Systems, **26** (2018), 43–65. 1

[21] C. G. Looney, *Fuzzy Petri nets for rule-based decision making*, IEEE Trans. Syst. Man Cybern., **18** (1988), 178–183. 1

[22] M. I. Mary Metilda, D. Lalitha, *Petri Nets for pasting tiles*, Intelligent Computing in Engineering, Springer, Singapore, (2020), 701–708. 1, 2.9

[23] M. I. Mary Metilda, D. Lalitha, *Kolam generated by color Petri Nets*, Information and Communication Technology for Sustainable Development, Springer, Singapore, (2020), 675–681. 1, 2.9

[24] C. Moraga, *Some properties of fuzzy languages*, Computational Intelligence, Theory and Applications. Springer, Berlin, Heidelberg, (2006), 367–374. 1, 2.10

[25] T. Muratma, *Perti nets: properties, analysis and applications*, Proceedings of IEEE, **77** (1989), 541–580. 1, 2

[26] J. N. Mordeson, D. S. Malik, *Fuzzy Automata and Languages: Theory and Applications*, CRC Press, (2002). 1, 2, 2.1, 2.2, 2.3, 2.4, 2.5

[27] C. A. Petri, *Communication with automata*, (1966). 1, 2

[28] P. Linz, *An introduction to formal languages and automata*, Jones & Bartlett Learning, (2006). 1, 2

[29] J. L. Peterson, *Petri Nets*, Comput. Surveys, **9** (1977), 223–252. 1

[30] J. L. Peterson, *Petri Net theory and the modeling of systems*, Prentice Hall PTR, (1981). 1, 2, 2.6, 2.7, 2.8

[31] G. Rozenberg (Ed.), *Advances in Petri Nets*, Springer Science & Business Media, (1987). 1

[32] D. K. Shirley Gloria, S. Devi, K. Nirmala, *Regular string-token Petri Nets*, Malaya J. Mat., **8** (2020), 445–449. 1, 2

[33] P. Usha, K. Thirusangu, B. Immanuel, *Tree-Token Petri Nets and Derivation Trees*, Ann. Pure Appl. Math., **8** (2014), 219–226. 1

[34] X. Yin, S. Lafortune, *On the decidability and complexity of diagnosability for labeled Petri Nets*, IIEEE Trans. Automat. Control, **62** (2017), 5931–5938. 1

[35] L. A. Zadeh, *Fuzzy sets*, Inf. Control., **8** (1965), 338-353. 1

[36] C. Zhang, G. Tian, A.M. Fathollahi-Fard, W. Wang, P. Wu, Z. Li, *Interval-valued intuitionistic uncertain linguistic cloud petri net and its application to risk assessment for subway fire accident*, IEEE Trans. Autom. Sci. Eng., **19** (2022), 163–177. 1

[37] K.-Q Zhou, A. M. Zain, *Fuzzy Petri Nets and industrial applications: a review*, Artif. Intell. Rev., **45** (2016), 405–446. 1