Journal of Algebra and Applied Mathematics
Vol. 21 (2023), No.1, pp.35-52
ISSN: 2319-7234

URL : www.sasip.net

Formulation of β-language from the semi-deterministic pushdown automata(SDPDA) of order n

A. Jain*, S. Jain, H. Ghazwani and G.C. Petalcorin, Jr.

Abstract. The authors of [4] introduced the notion of semi-deterministic pushdown automata (SDPDA) of order n as a midway stage between already known deterministic pushdown automata (DPDA) and nondeterministic pushdown automata (NPDA). The authors of [4] also introduced a new family of languages viz. β-languages of oder n that lies between deterministic context-free languages and nondeterministic context-free languages and have shown that given a β-language of order n, there exists an SDPDA of the same order that accepts exactly the given β-languages of order n. In this paper, we formulate an equivalent β-language from the language of an SDPDA order n.

AMS Subject Classification (2020): 68Q45

Keywords: Nondeterministic pushdown automata, deterministic pushdown automata, context-free grammar

1. Introduction

We know that the nondeterministic pushdown automata (NPDA) is the machine counterpart of context-free languages while deterministic pushdown automata (DPDA) can be associated with a subset of context-free languages viz. deterministic context-free langauges.

Motivated by the idea to have a pushdown automata with properties lying between that of DPDA and NPDA, the authors of [4] introduced the notion of semi-deterministic pushdown automata (SDPDA) of order

*Corresponding author

$n(n \geq 1)$ by controlling the choice of moves of an NPDA. The authors of [4] also introduced the notion of β-grammar and β-language of order n as a subclass of context-free grammar and context-free language respectively. The class of β-languages of order n lies between deterministic context-free languages and nondeterministic context-free languages. The authors of [4] have further shown that given a β-language of order n, there exists an SDPDA of the same order that accepts exactly the β-language of order n.

In this paper, we construct a β-language of order n from the language of a given SDPDA of the same order.

2. Preliminaries

We first informally define "semi-deterministic pushdown automata (SDPDA) of order n" as follows:

Definition 2.1 [4]. A "semi-deterministic pushdown automata(SDPDA) of order n" is a PDA that can make atmost "n"$(n \geq 1)$ transitions corresponding to a given input symbol (real or virtual input λ) and stack top symbol.

We now formally define a "semi-deterministic pushdown automata (SDPDA) of order n" as follows:

Definition 2.2 [4]. A "semi-deterministic pushdown automata(SDPDA) of order n" consists of

1. A finite set of states denoted by Q.
2. A finite set of input symbols including the empty string symbol λ. This is denoted by $\Sigma U\{\lambda\}$. Σ is called real input alphabet.
3. A finite set of stack symbols denoted by $\Gamma \cup \{\lambda\}$ where Γ is the real stack alphabet.

4. A transition function δ, that takes as arguments a state, an input symbol and stack top symbol and returns atmost n pairs of the form (q, x) where q is the next state of the control unit and x is a string of stack symbols which is put on the top of the stack in place of the single stack symbol there before with left most symbol of the string to be placed highest on the stack.

5. An initial start state $q_0 \in Q$.

6. A stack start symbol $Z_0 \in \Gamma$.

7. A set of final states $F \subseteq Q$.

Note that δ is defined in such a way that it needs a stack symbol and no move is possible if the stack is empty.

We can also denote an SDPDA of order n by a "sep-tuple" notation as

$$M(n) = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

where $M(n)$ is the name of the SDPDA of order n, Q is the finite set of internal states of the control unit, Σ is the set of real input symbols, Γ is the set of stack symbols, δ is the transition function from $Q \times (\Sigma U\{\lambda\}) \times \Gamma$ to finite subset of $Q \times \Gamma^*$ of order atmost n, $q_0 \in Q$ is the initial state of the control unit, $Z_0 \in \Gamma$ is the stack start symbol and $F \subseteq Q$ is the set of the final states.

Observations

(i) SDPDA (1) $\subseteq SDPDA(2) \subseteq SDPDA(3) \subseteq \cdots$.

(ii) For larger values of n, an SDPDA of order n behaves like an NDPDA.

We now define the "language accepted by an SDPDA of order n".

Definition 2.3 [4] (Language accepted by final state). Let $M(n) = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be an SDPDA of order n. The language accepted by $M(n)$ is the set

$$L(M(n)) = \{w \in \Sigma^* : (q_0, w, Z_0) \overset{*}{\vdash} (p, \lambda, u), p \in F, u \in \Gamma^*\}.$$

In other words, the language accepted by $M(n)$ is the set of all strings that can put $M(n)$ into a final state at the end of the input string irrespective of the final stack contents. This language is referred as the "language accepted by final state".

Another equivalent way of defining the language accepted by an SDPDA of order n is to be the set of all input strings for which some sequence of moves causes the SDPDA to empty its stack. This language is referred to as the "language accepted by empty stack" which we formally define as follows:

Definition 2.4 [4] (Language accepted by empty statck). For an SDPDA $M(n) = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, we define the "language accepted by empty stack" to be

$$L'(M(n)) = \{w : (q_0, w, Z_0) \overset{*}{\vdash} (p, \lambda, \lambda) \text{ for some } p \in Q\}.$$

The two definitions of acceptance are equivalent in the sense that if a language can be accepted by empty stack by some SDPDA of order n, then it can also be accepted by final state by some other SDPDA of order n, and vice versa.

Finally, the β-grammar and β-language of order n are defined as follows:

Definition 2.5 [4]. A context-free grammar $G = (V, T, S, P)$ is said to be a "β-grammar of order n"$(n \geq 1)$ if all productions in P are of the form

$$A \to ax$$

where $a \in T \cup \{\lambda\}$ and $x \in V^*$ and any pair (A, a) occurs atmost n times in P. A β-grammar of order n will be denoted by $\beta(n)$.

Definition 2.6 [4]. The language of $\beta(n)$ will be called as "β-language of order n".

Remark. β-grammar of order 1 viz. $\beta(1)$ is same as the simple grammar [1-3, 7].

The authors of [4] have shown that for every β-language of order n, there is an SDPDA of the same order that accepts it.

Theorem 2.7 [4]. *For any β-language of order n, there exists an SDPDA $M(n)$ of order n such that*

$$\beta(n) = L(M(n).$$

3. Construction of β-language from the language of an SDPDA of order n

In this section, we prove the main result pertaining to the construction of β-language (equivalently β-grammar) from the language of an SDPDA of order n. The result is further illustrated with the help of an example.

Theorem 3.1. If L is $L(M(n))$ for some SDPDA $M(n) = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \phi)$ of order n, then L is a β-language of order n.

Proof. The construction of the associated β-grammar is based on the fact that the contents of the stack are reflected in the variable part of the sentential form while the processed input is the terminal prefix of the sentential form. Without any loss of generality, we may assume that the SDPDA $M(n)$ accepts an input string iff the stack is empty after processing the string.

Now, let $G\ (V, T, P, S)$ be a β-grammar where

$V = \{S\} \cup \{[q, A, p] : q, p \in Q \text{ and } A \in \Gamma\}$

$=$ set of objects of the form $[q, A, p]$ where q and p are states in Q and A is a stack symbol in Γ plus the new plus the new symbol "S'' which is the start symbol;

$T = \Sigma$;

P is the set of productions given by

(i) for all states $p \in Q$, G has the production of the form

$$S \to [q_0, Z_0, p];$$

(ii) if $(\delta(q, a, A)$ contains the pair (r, λ), then the production rule is

$$[q, A, r] \to a;$$

(iii) if $\delta(q, a, A)$ contains the pair $(r, B_1 B_2 \cdots B_k)$ where $r \in Q$ and $B_1, B_2, \cdots, B_k \in \Gamma$, then the production rule is

$$[q, A, r_k] \to a[r, B_1, r_1][r_1, B_2, r_2] \cdots [r_{k-1}, B_k, r_k],$$

where $r_1, r_2, \cdots, r_k$ assume all possible values in Q.

The interpretation of the production rule in (iii) is that pop A and push $B_1 B_2 \cdots B_k$ on the stack and go from state q to state r while reading "a'' (which may be λ), then use some more input to pop B_1 off the stack while going from the state r to state r_1, then read some more input that pop B_2 off the stack and goes from state r_1 to r_2 and so on. The next effect of all these moves is popping A ad transition of states from state q to state r_k.

Clearly, G is a β-grammar of order t where $t \leq |Q|^k n$. However, after removing the useless variables and useless productions, we get the minimal equivalent β-grammar of order exactly equal to n.

We note that the production rules in G are defined so that

$$[q, A, p] \implies_G^* w \text{ for } w \in \Sigma^*$$

iff w causes $M(n)$ to erase A from its stack by some sequence of moves beginning in state q and ending in state p.

There might be some states "t'' that cannot be reached from state q while erasing A. In that case, the resulting variables $[q, A, t]$ are useless symbols and do not affect the language generated by the β-grammar. We shall remove such type of useless variables and productions involving such useless variables during minimization of the newly constructed β-grammar G.

Now, in order to show

$$L(G) = L(M(n)),$$

we prove by induction on the number of steps in the derivation of G or number of moves on $M(n)$ that

$$[q, A, p] \implies_G^* w \text{ for } w \in \Sigma^* \text{ iff } (q, w, A) \vdash_M^* (p, \lambda, \lambda).$$

(If). Firstly, we show by induction on the number of moves "i'' made by the SDPDA of order n that if

$$(q, w, q) \vdash_M^i (p, \lambda, \lambda),$$

then

$$[q, A, p] \implies_G^* w.$$

Basis. If $i = 1$, then w is either λ or a single real input symbol and $(p, \lambda) \in \delta(q, w, A)$.

Thus, by the construction of production rules of G, we have

$$[q, A, p] \longrightarrow w \text{ is a production of } G,$$

and so

$$[q, A, p] \Longrightarrow w.$$

Induction: Now, suppose $i > 1$. Let $w = au$ where $a \in \Sigma \cup \{\lambda\}, u \in \Sigma^*$ and $(q, w, A) = (q, au, A) \vdash_M (r_0, u, B_1, B_2 \cdots B_k) (\vdash_M^{i-1}) (p, \lambda, \lambda)$.

It follows that the pair $(r_0, B_1 B_2 \cdots B_k)$ must be in $\delta(q, a, A)$. Further, by the construction of production rules of G, there must be a production in G of the form

$$[q, A, r_k] \to a[r_0, B_1, r_1][r_1, B_2, r_2] \cdots [r_{k-1}, B_k, r_k], \tag{1}$$

where $r_k = p$ and $r_1, r_2, \cdots, r_{k-1}$ are any states in Q.

In particular, we may observe that each of the symbols $B_1, B_2, \cdots, B_k$ get popped off the stack in turn, and we may choose r_j to be the state of the SDPDA when B_j is popped off for $j = 1, 2, \cdots, k-1$. Let $w = w_1 w_2 \cdots w_k$, where w_j is the input consumed while B_j is popped off the stack. Then we know that

$$[r_{j-1}, w_j, B_j] \vdash^* (r_j, \lambda, \lambda)$$

in fewer than i moves.

We apply induction hypothesis and get

$$[r_{j-1}, B_j, r_j] \Longrightarrow_G^* w_j \text{ for } 1 \leq j \leq k-1. \tag{2}$$

We use the derivations give in (2) together with the first production

given in (1) to conclude the following:

$$\begin{aligned}[q, A, r_k] &\Longrightarrow a[r_0, B_1, r_1][r_1, B_2, r_2] \cdots [r_{k-1}, B_k, r_k] \\ &\Longrightarrow^* aw_1[r_1, B_2, r_2][r_2, B_3, r_3] \cdots [r_{k-1}, B_k, r_k] \\ &\Longrightarrow^* aw_1w_2[r_2, B_3, r_3][[r_3, B_4, r_4] \cdots [r_{k-1}, B_k, r_k] \\ &\Longrightarrow^* aw_1w_2 \cdots w_k = w,\end{aligned}$$

where $r_k = p$.

(Only-if). Conversely, suppose that

$$[q, A, p] \Longrightarrow_G^* w,$$

we show by induction on the number of steps "i" in the derivation of w that

$$(q, w, A) \vdash_M^* (p, \lambda, \lambda).$$

Basis. For $i = 1$, w is either λ or a symbol in Σ and $[q, A, p] \rightarrow w$ must be a production of G. The only way for this production to exist is in the case when there is a transition of $M(n)$ in which the symbol A is popped off the stack and state q becomes state p. That is (p, λ) must be in (q, a, A) and $a = w$. But then, we have

$$(q, w, A) \vdash_M (p, \lambda, \lambda).$$

Induction. Now, assume $i > 1$.

Suppose $[q, A, p] \Longrightarrow^*$ in i steps.

Consider the first sentential form explicitly which must look like

$$[q, A, r_k] \Longrightarrow a[r_0, B_1, r_1][r_1, B_2, r_2] \cdots [r_{k-1}, B_k, r_k] \Longrightarrow^{(i-1)} w, \qquad (3)$$

where $r_k = p$.

The production in (3) must come from the fact that $(r_0, B_1B_2 \cdots B_k)$ is in $\delta(q, a, A)$.

We can write w as $w = aw_1w_2 \cdots w_k$ such that

$$[r_{j-1}, B_j, r_k] \Longrightarrow^* w_j \text{ for } j = 1, 2, \cdots, k,$$

with each derivation taking fewer than i steps.

By the induction hypothesis, we get

$$(r_{j-1}, w_j, B_j) \vdash^*_M (r_j, \lambda, \lambda) \text{ for } 1 \leq j \leq k. \tag{4}$$

The sequence of $ID's$ in (4) clearly show that

$$(r_{j-1}, w_jw_{j+1} \cdots w_k, B_jB_{j+1} \cdots B_k) \vdash^* (r_j, w_{j+1} \cdots w_k, B_{j+1} \cdots B_k).$$

If we put all these sequences together, we see that

$$\begin{aligned}
(q, aw_1w_2 \cdots w_k, A) &\vdash (r_0, w_1w_2 \cdots w_k, B_1B_2 \cdots B_k) \\
&\vdash^* (r_1, w_2w_3 \cdots w_k, B_2B_3 \cdots B_k) \\
&\vdash^* (r_2, w_3w_4 \cdots w_k, B_3B_4 \cdots B_k) \\
&\vdash^* \cdots\cdots \\
&\vdash^* (r_k, \lambda, \lambda) \\
&= (p, \lambda, \lambda)
\end{aligned}$$

Thus, we have

$$(q, w, A) \Longrightarrow^* (p, \lambda, \lambda).$$

Combining the two cases, we have shown that

$$[q, A, p] \Longrightarrow^*_G w \text{ iff } (q, w, A) \vdash^*_M (p, \lambda, \lambda). \tag{5}$$

Now, $S \Longrightarrow^* w$ iff $[q_0, Z_0, p] \Longrightarrow^* w$ for some $p \in Q$, because of the way the rules for start symbol S are constructed.

Also, in view of (5), we have

$$[q_0, Z_0, p] \Longrightarrow_G^* w \text{ iff } (q, w, Z_0) \vdash_M^* (p, \lambda, \lambda),$$

iff $M(n)$ accepts w by empty stack.

Hence

$$w \in L(G) \text{ iff } w \in L(M(n));$$

or equivalently

$$L(G) = L(M(n)).$$

Now, the required minimal β-grammar of order n can be obtained from the β-grammar G by removing useless variables and useless productions in G. □

Example 3.2. Consider an SDPDA of order 2 that accepts the language $\{ww^R | w \in \{a, b\}^*\}$ by empty stack and is given by

$$M(2) = \{(q_0, q_1), \{a, b\}, \{Z_0, B, G\}, \delta, q_0, Z_0, \phi\},$$

where

$$\begin{aligned}
Q &= \{q, q_1\} = \text{set of states of SDPDA},\\
\Sigma &= \{a, b\} = \text{ set of real input symbols},\\
\Gamma &= \{Z_0, B, G\} = \text{set of real stack alphabet},\\
\delta &= \text{transition function},\\
q_0 &= \text{initial start state},\\
Z_0 &= \text{stack start symbol},
\end{aligned}$$

and δ is given by

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, BZ_0),\\
\delta(q_0, b, Z_0) &= (q_0, GZ_0),\\
\delta(q_0, a, B) &= \{(q_0, BB), (q_1, \lambda)\},\\
\delta(q_0, a, G) &= (q_0, BG)),\\
\delta(q_0, b, B) &= (q_0, GB),\\
\delta(q_0, b, G) &= \{(q_0, GG), (q_1, \lambda)\},\\
\delta(q_1, a, B) &= (q_1, \lambda),\\
\delta(q_1, b, G) &= (q_1, \lambda),\\
\delta(q_0, \lambda, Z_0) &= (q_1, \lambda),\\
\delta(q_1, \lambda, Z_0) &= (q_1, \lambda).
\end{aligned}$$

We obtain the following equivalent β-grammar G using the algorithm discussed in Theorem 3.1:

$$G = (V, T, P, S)$$

where

$$\begin{aligned}
V = \{&[q_0, Z_0, q_0], [q_), Z_0, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1],\\
&[q_0, B, q_0], [q_0, B, q_1], [q_1, B_0, q_0], [q_1, B, q_1],\\
&[q_0, G, q_0], [q_0, G, q_1], [q_1, G, q_0], [q_1, G_1, q_1], S\},\\
T = \Sigma = \{&a, b\},
\end{aligned}$$

and production rules in P are given by

$$\begin{aligned}
S &\longrightarrow [q_0, Z_0, q_0];\\
S &\longrightarrow [q_0, Z_0, q_1];\\
[q_0, Z_0, q_0] &\longrightarrow a[q_0, B, q_0][q_0, Z_0, q_0]; [\text{useless}]
\end{aligned}$$

$$[q_0, Z_0, q_0] \longrightarrow a[q_0, B, q_1][q_1, Z_0, q_0];$$

$$[q_0, Z_0, q_1] \longrightarrow a[q_0, B, q_0][q_0, Z_0, q_1]; [\text{useless}]$$

$$[q_0, Z_0, q_1] \longrightarrow a[q_0, B, q_1][q_1, Z_0, q_1];$$

$$[q_0, Z_0, q_0] \longrightarrow b[q_0, G, q_0][q_0, Z_0, q_0]; [\text{useless}]$$

$$[q_0, Z_0, q_0] \longrightarrow b[q_0, G, q_1][q_1, Z_0, q_0]; [\text{useless}]$$

$$[q_0, Z_0, q_1] \longrightarrow b[q_0, G, q_0][q_0, Z_0, q_1]; [\text{useless}]$$

$$[q_0, Z_0, q_1] \longrightarrow b[q_0, G, q_1][q_1, Z_0, q_1];$$

$$[q_0, B, q_0] \longrightarrow a[q_0, B, q_0][q_0, B, q_0]; [\text{useless}]$$

$$[q_0, B, q_0] \longrightarrow a[q_0, B, q_1][q_1, B, q_0]; [\text{useless}]$$

$$[q_0, B, q_1] \longrightarrow a[q_0, B, q_0][q_0, B, q_1]; [\text{useless}]$$

$$[q_0, B, q_1] \longrightarrow a[q_0, B, q_1][q_1, B, q_1];$$

$$[q_0, B, q_1] \longrightarrow a;$$

$$[q_0, G, q_0] \longrightarrow a[q_0, B, q_0][q_0, G, q_0]; [\text{useless}]$$

$$[q_0, G, q_0] \longrightarrow a[q_0, B, q_1][q_1, G, q_0]; [\text{useless}]$$

$$[q_0, G, q_1] \longrightarrow a[q_0, B, q_0][q_0, G, q_1]; [\text{useless}]$$

$$[q_0, G, q_1] \longrightarrow a[q_0, B, q_1][q_0, G, q_1];$$

$$[q_0, B, q_0] \longrightarrow b[q_0, G, q_0][q_0, B, q_0]; [\text{useless}]$$

$$[q_0, B, q_0] \longrightarrow b[q_0, G, q_1][q_1, B, q_0]; [\text{useless}]$$

$$[q_0, B, q_1] \longrightarrow b[q_0, G, q_0][q_0, B, q_1];$$

$$[q_0, B, q_1] \longrightarrow b[q_0, G, q_1][q_1, B, q_1];$$

$$[q_0, G, q_0] \longrightarrow b[q_0, G, q_0][q_0, G, q_0]; [\text{useless}]$$

$$[q_0, G, q_0] \longrightarrow b[q_0, G, q_1][q_1, G, q_0]; [\text{useless}]$$

$$[q_0, G, q_1] \longrightarrow b[q_0, G, q_0][q_0, G, q_1]; [\text{useless}]$$

$$[q_0, G, q_1] \longrightarrow b[q_0, G, q_1][q_1, G, q_1];$$

$$[q_0, G, q_1] \longrightarrow b;$$
$$[q_1, B, q_1] \longrightarrow a;$$
$$[q_0, Z_0, q_1] \longrightarrow \lambda;$$
$$[q_1, Z_0, q_1] \longrightarrow \lambda.$$

Eliminating the useless productions as marked above and also renaming the variables in a user friendly form as given below:

$$[q_0, Z_0, q_0] = L,$$
$$[q_0, Z_0, q_1] = M,$$
$$[q_0, B, q_0] = P,$$
$$[q_0, B, q_1] = Q,$$
$$[q_0, G, q_0] = R,$$
$$[q_0, G, q_1] = T,$$
$$[q_1, Z_0, q_1] = U,$$
$$[q_1, B, q_1] = V,$$
$$[q_1, G, q_1] = W,$$

we get the following simplified β-grammar:

$$S \longrightarrow L;$$
$$S \longrightarrow M;$$
$$L \longrightarrow aQU;$$
$$M \longrightarrow aQU;$$
$$M \longrightarrow bTU;$$
$$Q \longrightarrow aQV;$$
$$T \longrightarrow aQW;$$

$$Q \longrightarrow bTQ;$$
$$Q \longrightarrow bTV;$$
$$T \longrightarrow bTW;$$
$$T \longrightarrow b;$$
$$V \longrightarrow a;$$
$$W \longrightarrow b;$$
$$M \longrightarrow \lambda;$$
$$U \longrightarrow \lambda.$$

To obtain the minimal equivalent β-grammar of order 2, we identify

$$L = M = U, \quad V = Q \quad \text{and } T = W,$$

and obtain the following simplified minimal equivalent β-grammar of order 2:

$$S \longrightarrow M;$$
$$M \longrightarrow aQM;$$
$$M \longrightarrow bTM;$$
$$Q \longrightarrow aQQ;$$
$$Q \longrightarrow a;$$
$$Q \longrightarrow bTQ;$$
$$T \longrightarrow aQT;$$
$$T \longrightarrow bTT;$$
$$T \longrightarrow b;$$
$$M \longrightarrow \lambda.$$

Illustration. Consider the derivation of string $abaaba$:

$S \to M \to aQM \to abTQM \to abaQTQM \to abaaTQM \to abaabQM$
$\to abaabaM \to abaaba.$

4. Conclusion

In this paper, we have discussed an induction based algorithmic method to construct a β-grammar (equivalently β-language) from the language of a given SDPDA of order n. The initial order of the constructed β-grammar depends on the number of states as well as the order "n'' of the given SDPDA. But after removing useless variables and useless productions, we obtain the minimal equivalent β-grammar of order exactly equal to the order of the language of the given SDPDA.

References

[1] A.V. Aho and J.D. Ullman, The Theory of Parsing, Translation and Computing, Vol. 1, Englewood Cliffs. N.J.: Prentice Hall, 1972.

[2] M.A. Harriosn, Introduction to Fromal languages Theory, Addison Wesley, Reading, Mass., 1978.

[3] J.E. Hopcroft, R. Motwani and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Pearson Education, Addison-Wesley, Reading, Mass., 2004.

[4] A. Jain, G.C. Petalcorin, Jr. and K.-S. Lee, *Semi-deterministic pushdown automata (SDPDA) of order "n" and β-languages*, J. Algeb. and Applied Mathematics, 14 (2016), 27-40.

[5] A. Jain, K.P. Shum, G.C. Petalcorin, Jr. and K.-S. Lee, *α-grammar and quasi-deterministic pushdown automta (QDPDA) of order "n"*, J. Algeb. and Applied Mathematics, 18 (2020), 99-114.

[6] A. Jain, S. Jain and G.C. Petalcorin, Jr.: *Construction of α-language from the language of a QDPDA of order "n"*, J. Anal. and Appl., 20 (2022), 135-150.

[7] P. Linz, An Introduction to Formal lanuages and Automta, Narosa Publishing House, 2009.

[8] I. Petre and S. Arto, *Algebraic systems and pushdown automata*, In M. Drosde, W. Kuick and H. Vogler, Ed., Handbook of Weighted Automata, Springer, Chapter 7, 2009, pp.257-289.

[9] G.E. Revesz, Introduction to Formal Languages, McGraw-Hill, 1983.

[10] A. Salomaa, *Computations and Automata*, in Encyclopedia of Mathematics and Its Applications, Cambridge University Press, Ca,mbridge, 1985.

[11] W. Kuich and S. Arto, Semirings, automata, languages, Springer Verlag, 1986.

Department of Mathematics
Shanxi Normal University
P.R. China
E-mail: jainarihant@gmx.com

Department of Mathematics
Shanxi Normal University
P.R. China
E-mail: sapnajain@gmx.com

Department of Mathematics
Science College
Jazan university
Kingdom of Saudi Arabia
E-mail: hqghazwani@jazanu.edu.sa

Department of Mathematics and Statistics
College of Science and Mathematics
MSU-Iligan Institute of Technology
Tibanga, Iligan City
Philippines
E-mail: gaudencio.petalcorin@g.msuiit.edu.ph
(*Received: December, 2022; Revised: January, 2023*)