Acta Cybernetica **00** (0000) 1–16.

On Pure Multi-Pushdown Automata that Perform Complete Pushdown Pops

Tomáš Masopust* and Alexander Meduna*

Abstract

This paper introduces and discusses pure multi-pushdown automata that remove symbols from their pushdowns only by performing complete pushdown pops. This means that during a pop operation, the entire pushdown is compared with a prefix of the input, and if they match, the whole contents of the pushdown is erased and the input is advanced by the prefix. The paper proves that these automata define an infinite hierarchy of language families identical with the infinite hierarchy of language families resulting from right linear simple matrix grammars. In addition, this paper discusses some other extensions of these automata with respect to operations they can perform with their pushdowns. More specifically, it discusses pure multi-pushdown automata that perform complete pushdown pops that are allowed to join two pushdowns and/or create a new pushdown.

Keywords: Pure multi-pushdown automaton, complete pushdown pop, infinite hierarchy.

1 Introduction

Indisputably, *pushdown automata* fulfill a crucial role in formal language theory. Therefore, it comes as no surprise that this theory has introduced many variants of these automata (consult, for example, [1, 5, 6, 7, 8, 11, 12, 14, 17, 18] for more details).

It is well-known that the family of languages accepted by pushdown automata (with only one pushdown) coincides with the family of context-free languages and that adding any more pushdown makes these automata as powerful as Turing machines. Considering the pushdown alphabet, it is not hard to see that any number of pushdown symbols can be encoded by two different pushdown symbols. However, if the pushdown alphabet is a singleton (more precisely, we have one pushdown symbol A, and a bottom-of-pushdown symbol Z, $Z \neq A$, Z appears only on the bottom of the pushdown), we obtain so-called *counter automata* or *counter machines*. It is known that these automata accept languages from a proper subfamily of the family of context-free languages if they are equipped with only one counter, or the family of recursively enumerable languages if they are equipped

*Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 61266, Czech Republic, E-mail: `tomas.masopust@mail.muni.cz, meduna@fit.vutbr.cz`.

with two or more counters (see [9]). Furthermore, the pushdown alphabet can, in general, contain symbols that are not in the input alphabet, i.e., symbols that will never occur as a part of the input. If the pushdown automata are restricted so that the pushdown alphabet does not contain any such symbols, we obtain so-called *pure pushdown automata*. Clearly, with respect to the cardinality of the input alphabet, pure pushdown automata with only one input symbol are as powerful as counter automata, whereas with two or more different input symbols they are as powerful as pushdown automata. Therefore, the family of languages accepted by pure pushdown automata with only one pushdown coincides either with the family of languages accepted by one-counter automata, or with the family of context-free languages. Finally, it immediately follows from the previous explanations that pure pushdown automata with two or more pushdowns are as powerful as Turing machines. For an overview of multi-pushdown automata see the paper by Fischer [4] and the references therein.

The present paper continues the investigations in this classical topic of formal language theory. More specifically, it discusses pure multi-pushdown automata that can remove symbols from their pushdowns only by performing a complete pushdown pop. This means that during a pop operation, the entire pushdown is compared with a prefix of the input, and if they match, the whole contents of the pushdown is eliminated and, simultaneously, the input is advanced by the prefix. This paper demonstrates that these automata define an infinite hierarchy of language families identical with the infinite hierarchy of language families resulting from the following grammars and automata:

1. equal matrix languages (see Siromoney [16]);
2. right linear simple matrix grammars (see Ibarra [10]);
3. multi-tape one-way non-writing automata (see Fischer and Rosenberg [5]);
4. finite-turn checking automata (see Siromoney [17]);
5. all-move self-regulating finite automata (see Meduna and Masopust [13]).

In addition, this paper discusses pure multi-pushdown automata that perform complete pushdown pops that are allowed (in some sense) to join two pushdowns and/or introduce a new pushdown. These operations imply another infinite hierarchy of language families dependent upon the number of pushdowns.

In its conclusion, this paper formulates some open problems.

2 Preliminaries and Definitions

In this paper, we assume that the reader is familiar with the theory of automata and formal languages (see [15]). For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$. For $w \in V^*$ and $W \subseteq V$, w^R denotes the mirror image of w and $occur(w,W)$ denotes the number of occurrences of symbols from W in w. Let $\mathscr{L}_{REG}$ denote the family of regular languages.

A *context-free grammar* is a quadruple $G = (N, T, P, S)$, where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, and P is a finite set of productions of the form $A \rightarrow v$, where $A \in N$ and $v \in V^*$.

In what follows, productions from P are labeled by elements of a finite set Q chosen so that there is a bijection lab from P to Q. Then, $Q = lab(P) = \{lab(p) : p \in P\}$ is said to be a set of *production labels*. For the brevity, we hereafter write $q : A \rightarrow v \in P$ instead of $A \rightarrow v \in P$ with $lab(A \rightarrow v) = q$.

Let $q : A \rightarrow v \in P$ and $x, y \in V^*$. Then, G makes a derivation step from xAy to xvy, written as $xAy \Rightarrow xvy$. In the standard way, we define $\Rightarrow^m$, for $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. To express that G performs $x \Rightarrow^m y$, for some $x, y \in V^*$, by using a sequence of productions $q_1, q_2, \ldots, q_m$, we write $x \Rightarrow^m y \ [q_1 q_2 \ldots q_m]$. The language generated by a context-free grammar G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$ and is said to be a *context-free language*. The family of all context-free languages is denoted by $\mathscr{L}_{CF}$.

For $n \geq 1$, an *n-right linear simple matrix grammar* (defined in [16] (as equal matrix grammars) and in [10], see also [19]) is an $(n+3)$-tuple $G = (N_1, N_2, \ldots, N_n, T, P, S)$, where $N_1, N_2, \ldots, N_n$ are pairwise disjoint nonterminal alphabets, T is a terminal alphabet, $N = N_1 \cup N_2 \cup \cdots \cup N_n$, $S \notin N \cup T$ is the start symbol, $N \cap T = \emptyset$, and P is a finite set of matrix productions of the following three forms:

1. $[S \rightarrow X_1 X_2 \ldots X_n]$, $\quad X_i \in N_i$, $1 \leq i \leq n$;
2. $[X_1 \rightarrow w_1 Y_1, X_2 \rightarrow w_2 Y_2, \ldots, X_n \rightarrow w_n Y_n]$, $\quad w_i \in T^*$, $X_i, Y_i \in N_i$, $1 \leq i \leq n$;
3. $[X_1 \rightarrow w_1, X_2 \rightarrow w_2, \ldots, X_n \rightarrow w_n]$, $\quad X_i \in N_i$, $w_i \in T^*$, $1 \leq i \leq n$.

For $x, y \in (N \cup T \cup \{S\})^*$, $x \Rightarrow y$ provided that

1. either $x = S$ and $[S \rightarrow y] \in P$,
2. or $x = y_1 X_1 y_2 X_2 \ldots y_n X_n$, $y = y_1 x_1 y_2 x_2 \ldots y_n x_n$, and $[X_1 \rightarrow x_1, \ldots, X_n \rightarrow x_n] \in P$.

As usual, we define $\Rightarrow^m$, for $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by an n-right linear simple matrix grammar G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$ and is said to be an *n-right linear simple matrix language*. The family of all n-right linear simple matrix languages is denoted by $\mathscr{L}_R^n$.

A *programmed grammar* is a quadruple $G = (N, T, P, S)$, where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, and P is a finite set of productions of the form $(q : A \rightarrow v, g(q))$, where $q : A \rightarrow v$ is a labeled context-free production and $g(q) \subseteq lab(P)$.

In every derivation of G, any two consecutive steps, $x \Rightarrow y \Rightarrow z$, made by productions $(p : A \rightarrow u, g(p))$ and $(q : B \rightarrow v, g(q))$, respectively, satisfy $q \in g(p)$. As usual, we define $\Rightarrow^m$, for $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a programmed grammar G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$ and is said to be a *programmed language*. The family of all programmed languages is denoted by $\mathscr{L}_P$.

Let D be a derivation of a string $w \in V^*$ in G of the form $w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_r$, for some $r \geq 1$, where $S = w_1$ and $w_r = w$. Set $Ind(D, G) = \max\{occur(w_i, N) : 1 \leq i \leq r\}$. For $w \in T^*$, set $Ind(w, G) = \min\{Ind(D, G) : D \text{ is a derivation of } w \text{ in } G\}$. The *index* of G is defined as $Ind(G) = \max\{Ind(w, G) : w \in L(G)\}$.

For $L \in \mathscr{L}_P$, set $Ind(L) = \min\{Ind(G) : L(G) = L,\ G$ is a programmed grammar$\}$. Finally, let $\mathscr{L}_P^n = \{L \in \mathscr{L}_P : Ind(L) \leq n\}$, for all $n \geq 1$, denote the family of all *programmed languages of index n*.

2.1 Pure Multi-Pushdown Automata that Perform Complete Pushdown Pops

Let n be a positive integer. A *pure n-pushdown automaton that performs complete pushdown pops*, an nPPDA for short, is a quadruple

$$M = (Q, T, R, s)\,,$$

where Q is a finite set of states, T is an alphabet of input symbols, $R \subseteq \mathscr{S} \times \mathscr{S}$ is a set of rules, $\mathscr{S} = \mathscr{S}_1 \cup \mathscr{S}_2 \cup \mathscr{S}_3 \cup \mathscr{S}_4$,

- $\mathscr{S}_1 = \{\langle q, pop\rangle : q \in Q\}$
- $\mathscr{S}_2 = \{\langle q, push, i, a\rangle : q \in Q,\ 1 \leq i \leq n,\ a \in T \cup \{\varepsilon\}\}$
- $\mathscr{S}_3 = \{\langle q, new, i\rangle : q \in Q,\ 1 \leq i \leq n\}$
- $\mathscr{S}_4 = \{\langle q, join, i\rangle : q \in Q,\ 2 \leq i \leq n\}$

and $s \notin \mathscr{S}$ is the start state. In what follows, we use the notation $p \to q$ for $(p, q) \in R$.

A configuration of M is a string over

$$(T^*\{\$\} \cup \{\varepsilon\})^n \times (\mathscr{S} \cup \{s\}) \times T^*\,.$$

Let $1 \leq k \leq n$ and $p \to q \in R$. We define the relation $\Rightarrow$ depending on the left-hand side of $p \to q$, i.e., p, as follows:

1. $\$^n sw \Rightarrow \$^n qw$, for $p = s$;
2. $w_k\$\ldots\$w_2\$w_1\$pw_1^R w \Rightarrow w_k\$\ldots\$w_2\$qw$ for $p = \langle r, pop\rangle$;
3. $w_k\$\ldots\$w_i\$\ldots\$w_1\$pw \Rightarrow w_k\$\ldots\$w_ia\$\ldots\$w_1\$qw$, for $p = \langle r, push, i, a\rangle$ and $i \leq k$;
4. $w_k\$\ldots\$w_i\$\ldots\$w_1\$pw \Rightarrow w_k\$\ldots\$w_i\$\$\ldots\$w_1\$qw$, for $p = \langle r, new, i\rangle$ and $i \leq k < n$;
5. $w_k\$\ldots\$w_1\$pw \Rightarrow \$w_k\$\ldots\$w_1\$qw$, for $p = \langle r, new, k+1\rangle$ and $k < n$;
6. $w_k\$\ldots\$w_i\$w_{i-1}\$\ldots\$w_1\$pw \Rightarrow w_k\$\ldots\$w_iw_{i-1}\$\ldots\$w_1\$qw$, for $p = \langle r, join, i\rangle$ and $i \leq k$.

Remark 1. Note that symbols \$ denote the tops of M's pushdowns and that the automaton cannot make a computational step unless there is at least one pushdown.

In the standard way, we define $\Rightarrow^m$, for $m \geq 0$, and $\Rightarrow^*$. Then, the language of an nPPDA M is defined as

$$L(M) = \{w \in T^* : \$^n sw \Rightarrow^* q, \text{ for some } q \in \mathscr{S}\},$$

where $\$^n sw \Rightarrow^* q$ is said to be a *successful computation of M on w*.

Finally, for $I \subseteq \{1,2,3,4\}$,

$$\mathscr{L}_I^n = \left\{L(M) : M = (Q,T,R,s) \text{ is an } n\text{PPDA with } R \subseteq \bigcup_{i\in I} \mathscr{S}_i \times \bigcup_{i\in I} \mathscr{S}_i \right\}.$$

3 Main Results

In this section, we demonstrate two infinite language hierarchies generated by pure multi-pushdown automata that perform complete pushdown pops according to their pushdown operations and the number of pushdowns. First, however, we generalize these automata so that they are allowed to push a string to their pushdowns in one computational step instead of only one symbol or the empty string.

3.1 Generalized *n*PPDAs

A *generalized nPPDA* is an nPPDA $M = (Q,T,R,s)$ with $R \subseteq \mathscr{S}' \times \mathscr{S}'$, where $\mathscr{S}'$ is a finite subset of $\mathscr{S}_1 \cup \mathscr{S}_2' \cup \mathscr{S}_3 \cup \mathscr{S}_4$; sets $\mathscr{S}_1$, $\mathscr{S}_3$, $\mathscr{S}_4$ are as in the case of standard nPPDA, and $\mathscr{S}_2$ is modified so that $a \in T$ is replaced with $u \in T^*$:

- $\mathscr{S}_2' = \{\langle q, push, i, u\rangle : q \in Q,\, 1 \leq i \leq n,\, u \in T^*\}$.

Correspondingly, the computational step is modified as follows:

3. $w_k\$\ldots\$w_i\$\ldots\$w_1\$\langle p, push, i, u\rangle w \Rightarrow w_k\$\ldots\$w_iu\$\ldots\$w_1\$qw$, for $i \leq k$.

The other computational steps are defined as in the case of standard nPPDA.

First, we prove that this generalization has no effect to the acceptance power of these automata.

Lemma 1. *Let M be a generalized nPPDA, for some $n \geq 1$. Then, there is an nPPDA, M', such that $L(M) = L(M')$.*

Informally, what M does in one derivation step, M' does in the-length-of-the-added-string steps.

Proof. Let $M = (Q,T,R,s)$ be a generalized nPPDA. Construct the following nPPDA $M' = (Q',T,R',s)$ by the following algorithm ($\mathscr{S}$ is as in the definition in Section 2.1):

1. Set $R' = \{p \to q \in R : p,q \in \mathscr{S} \cup \{s\}\}$ and $Q' = Q$;
2. For each $p \to \langle q, push, i, a_1a_2\ldots a_k\rangle \in R$ with $a_i \in T$, for $i = 1,\ldots,k$, $k \geq 2$, add
   a) states $q^{i,1}_{a_1a_2\ldots a_k}, q^{i,2}_{a_1a_2\ldots a_k}, \ldots, q^{i,k}_{a_1a_2\ldots a_k}$ to Q';

b) $p \to \langle q^{i,1}_{a_1a_2\dots a_k}, push, i, a_1\rangle$ to R';

c) $\langle q^{i,j}_{a_1a_2\dots a_k}, push, i, a_j\rangle \to \langle q^{i,j+1}_{a_1\dots a_k}, push, i, a_{j+1}\rangle$ to R', for $j = 1,\dots,k-1$;

d) for each $\langle q, push, i, a_1a_2\dots a_k\rangle \to r \in R$, add
$\langle q^{i,k}_{a_1a_2\dots a_k}, push, i, a_k\rangle \to r$ to $\begin{cases} R' & \text{for } r \in \mathscr{S}, \\ R & \text{otherwise.} \end{cases}$

3. If R' has been changed, then go to step 2.

It is not hard to see that $L(M) = L(M')$. □

3.2 Language Families

Consider an arbitrary $I \subseteq \{1,2,3,4\}$. It is not hard to see that if $1 \notin I$, then $\mathscr{L}^n_I = \emptyset$; such an automaton cannot remove \$s from its configurations. Furthermore, if $1 \in I$ and $2 \notin I$, then $\mathscr{L}^n_I = \{\varepsilon\}$; of course, such an automaton can remove all symbols \$ but cannot read any nonempty input. Thus, there are only four sets of interest: $\{1,2\}$, $\{1,2,3\}$, $\{1,2,4\}$, $\{1,2,3,4\}$. The following two lemmas are obvious.

Lemma 2. *For all $n \geq 1$,*

1. $\mathscr{L}^n_{\{1,2\}} \subseteq \mathscr{L}^n_{\{1,2,3\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$,
2. $\mathscr{L}^n_{\{1,2\}} \subseteq \mathscr{L}^n_{\{1,2,4\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$.

Lemma 3. $\mathscr{L}^1_{\{1,2\}} = \mathscr{L}^1_{\{1,2,3,4\}} = \mathscr{L}_{REG}$.

Now, consider an automaton with pop, push, and join operations. We will show how the join operation can be simulated by only push and pop operations without any change of the accepted language. Notice that the join operation applied to the ith pushdown appends the content of the ith pushdown to the bottom of the $(i-1)$st pushdown. Thus, to push a symbol to the jth pushdown in this automaton, for some $j \geq i$, equals to skipping the join operation and pushing the symbol to the $(j+1)$st pushdown. This is generalized and done by a sequence of the form $i_1i_2\dots i_m$ added to states, for some $m \leq n$, where $i_k \in \{1,0\}$, for $k = 1,\dots,m$, and $i_k = 0$ if and only if the i_kth pushdown has been joined. Then, the automaton starts with a sequence of n 1s, $11\dots1$, in its start state, and to push a symbol to the ith pushdown means to push the symbol to the lth pushdown, where l is the position of the ith 1 in the sequence from the left. Analogously, to make the pop operation, say from a state with $10\dots0i_l\dots i_k$, where $2 \leq l \leq k$ and $i_l = 1$, the new automaton makes $l-1$ pop operations and goes to a state with $i_l\dots i_k$. Finally, to join the ith pushdown means to replace the ith 1 with 0 in the state by the push operation pushing ε to the first pushdown.

Hence, we have the following lemma.

Lemma 4. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} = \mathscr{L}^n_{\{1,2,4\}}$.*

Corollary 1. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} = \mathscr{L}^n_{\{1,2,4\}} \subseteq \mathscr{L}^n_{\{1,2,3\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$.*

As far as the $\mathscr{L}^n_{\{1,2,3\}}$ language families are concerned, $n \geq 2$, we only know the following result.

Theorem 1. *For all $n \geq 2$, $\mathscr{L}^n_{\{1,2\}} \subset \mathscr{L}^n_{\{1,2,3\}}$.*

Proof. Let $L = \{a^k b^k : k \geq 1\}$. Clearly, $L \in \mathscr{L}^2_{\{1,2\}}$. Ibarra [10, Theorem 4.7] showed that $L^* \notin \mathscr{L}^m_{\{1,2\}}$, for $m \geq 1$. To prove this theorem, we show that $L^* \in \mathscr{L}^2_{\{1,2,3\}}$.

The automaton starts with the initial configuration $\$\$sa^kb^kw$, for some $w \in L^*$. Then, it simultaneously generates a^k and b^k in the first and the second pushdown, respectively, i.e., the configuration is of the form $b^k\$a^k\$pa^kb^kw$, for some $p \in \mathscr{S}$. Then, the automaton pops the first pushdown (reading a^k from the input) and creates a new one on the second position, i.e., its configuration is $\$b^k\$qb^kw$, for some $q \in \mathscr{S}$. Then, it pops the first pushdown again (reading b^k from the input) and creates a new pushdown, i.e., the configuration is $\$\$rw$, for some $r \in \mathscr{S}$. Thus, the cycle can be repeated. The formal proof is left to the reader. $\square$

Corollary 2. $\mathscr{L}^2_{\{1,2,3\}} - \bigcup_{n=1}^{\infty} \mathscr{L}^n_{\{1,2\}} \neq \emptyset$.

In the following two sections, language families $\mathscr{L}^n_{\{1,2\}}$ and $\mathscr{L}^n_{\{1,2,3,4\}}$ are discussed. Note that most of the questions concerning language families $\mathscr{L}^n_{\{1,2,3\}}$ are open (see the last section for more details).

3.3 Language Families $\mathscr{L}^n_{\{1,2\}}$

First, let us give an example. Note that in case $n = 2$, this example shows that the language, L, from the proof of Theorem 1 is in $\mathscr{L}^2_{\{1,2\}}$ as stated there.

Example 1. Consider an nPPDA $M = (\{s,q\}, \{a_1, a_2, \ldots, a_n\}, R, s)$ with R having the following rules:

1. $s \to \langle q, push, 1, a_1\rangle$,
2. $\langle q, push, i, a_i\rangle \to \langle q, push, i+1, a_{i+1}\rangle$, for $i = 1, \ldots, n-1$,
3. $\langle q, push, n, a_n\rangle \to \langle q, push, 1, a_1\rangle$,
4. $\langle q, push, n, a_n\rangle \to \langle q, pop\rangle$,
5. $\langle q, pop\rangle \to \langle q, pop\rangle$.

Then, $L(M) = \{a_1^k a_2^k \ldots a_n^k : k \geq 1\}$. $\square$

In the following, we prove that the power of nPPDAs with push and pop operations is precisely the power of n-right linear simple matrix grammars. First, however, notice that any such automaton, M, has the property that there is precisely n pop operations in any of its successful computations; clearly, the automaton has to pop n pushdowns and no new pushdown can be created. Moreover, we can prove that there is an equivalent automaton, M', such that in any successful computation of M', no pop operation precedes a push operation. To show this, let M' simulate M but if M pops the pushdown, M' skips the pop operation and increases the number of pop operations skipped so far recorded in its state. Thus, in any time, M' knows the number of pop operations applied in the corresponding computation of M, say k, $0 \leq k \leq n$. Then, if M pushes a symbol to the ith pushdown, M' pushes this symbol to the $(i+k)$th pushdown. Clearly, M' finishes (pops all its pushdowns one by one) only if M has performed n pop operations.

Lemma 5. *Let $n \geq 1$ and $L \in \mathscr{L}^n_{\{1,2\}}$. Then, there is an nPPDA, M, such that $L(M) = L$ and its sequence of operations applied during any successful computation, starting from s, is of the form*

$$s, push_1, push_2, \ldots, push_k, pop_1, pop_2, \ldots, pop_n$$

for some $k \geq 1$, $push_i \in \mathscr{S}_2$, for all $i = 1, \ldots, k$, and $pop_j \in \mathscr{S}_1$, for all $j = 1, \ldots, n$.

Proof. This immediately follows from the previous arguments and the fact that if there is no push operation in the successful computation, then we can push ε to the first pushdown, i.e., for some state t, $push_1 = \langle t, push, 1, \varepsilon \rangle$. □

Lemma 6. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} \subseteq \mathscr{L}^n_R$.*

Proof. Let $M = (Q, T, R, s)$ be an nPPDA with $R \subseteq (\mathscr{S}_1 \cup \mathscr{S}_2) \times (\mathscr{S}_1 \cup \mathscr{S}_2)$ satisfying the condition from Lemma 5. Clearly, without loss of generality, we can assume that $pop_1 = pop_2 = \cdots = pop_n = \langle r, pop \rangle$, for some $r \in Q$. Thus, $\mathscr{S}_1 = \{\langle r, pop \rangle\}$.

Let $G = (N_1, \ldots, N_n, T, P, S_G)$ and set $N_i = (\mathscr{S}_1 \cup \mathscr{S}_2) \times \{i\}$, for all $i = 1, \ldots, n$. Set $P = \{S_G \to \langle\langle r, pop \rangle, 1\rangle\langle\langle r, pop \rangle, 2\rangle \ldots \langle\langle r, pop \rangle, n\rangle : \langle r, pop \rangle \in \mathscr{S}_1\}$.

If $q \to p \in R$ is of the form

1. $\langle t, push, i, a \rangle \to \langle r, pop \rangle$, add
   $[\langle\langle r, pop \rangle, 1\rangle \to \langle q, 1\rangle, \ldots, \langle\langle r, pop \rangle, i\rangle \to a\langle q, i\rangle, \ldots, \langle\langle r, pop \rangle, n\rangle \to \langle q, n\rangle]$ to P;
2. $\langle r, push, i, a \rangle \to \langle t, push, j, b \rangle$, add
   $[\langle p, 1\rangle \to \langle q, 1\rangle, \ldots, \langle p, i\rangle \to a\langle q, i\rangle, \ldots, \langle p, n\rangle \to \langle q, n\rangle]$ to P;
3. $s \to p$, add
   $[\langle p, 1\rangle \to \varepsilon, \ldots, \langle p, i\rangle \to \varepsilon, \ldots, \langle p, n\rangle \to \varepsilon]$ to P.

Note that M starts with $s \to p$, continues with $p \to q$ followed by the application of a rule of the form $p \to \langle r, pop \rangle$, for some $p, q \in \mathscr{S}_2$, and finishes with $\langle r, pop \rangle \to \langle r, pop \rangle$ applied n-times. Denote the sequence of applied rules by $s, p_1, \ldots, p_k, pop_1, \ldots, pop_n$, for some $k \geq 1$. Then, G simulates M by the following sequence of productions: the initial production (simulating all n pop operations) followed by a sequence of productions $p'_k, \ldots, p'_1, s'$, where p'_k is constructed from p_k as in 1, p'_i from p_i as in 2, for all $i = 1, \ldots, k-1$, and s' from s as in 3. □

Lemma 7. *For all $n \geq 1$, $\mathscr{L}^n_R \subseteq \mathscr{L}^n_{\{1,2\}}$.*

Proof. Let $n \geq 1$ and $G = (N_1, \ldots, N_n, T, P, S)$ be an n-right linear simple matrix grammar. Construct the following generalized nPPDA $M = (Q, T, R, s)$, where $Q = \{(x, m) : x \in N_1 \ldots N_n, m \in P\} \cup \{S\}$ and R is defined as follows:

1. For $\alpha = X_1 \ldots X_n \in N_1 \ldots N_n$ and $m = [X_1 \to w_1, \ldots, X_n \to w_n] \in P$ with $w_i \in T^*$, for all $i = 1, 2, \ldots, n$, add $s \to \langle (\alpha, m), push, 1, w_1^R \rangle$ to R;

2. For $\alpha = X_1 \dots X_n$, $\beta = Y_1 \dots Y_n \in N_1 \dots N_n$, and $m' = [Y_1 \rightarrow v_1X_1, \dots, Y_n \rightarrow v_nX_n] \in P$, add $\langle(\alpha,m),push,n,w_n^R\rangle \rightarrow \langle(\beta,m'),push,1,v_1^R\rangle$ to R;
3. For $\alpha = Y_1 \dots Y_n \in N_1 \dots N_n$ and $m[i+1] = Y_{i+1} \rightarrow v_{i+1}X_{i+1}$ ($m[i]$ denotes the ith element of m) with $v_{i+1} \in T^*$ and $X_{i+1} \in N \cup \{\varepsilon\}$, for all $i = 1,\dots,n-1$, add $\langle(\alpha,m),push,i,v_i^R\rangle \rightarrow \langle(\alpha,m),push,i+1,v_{i+1}^R\rangle$ to R;
4. For $\alpha = X_1 \dots X_n$, if there is $[S \rightarrow X_1 \dots X_n] \in P$, add $\langle(\alpha,m),push,n,v_n^R\rangle \rightarrow \langle S,pop\rangle$ to R;
5. Add $\langle S,pop\rangle \rightarrow \langle S,pop\rangle$ to R.

Clearly, M simulates the derivation of G bottom-up and what G does in one derivation step, M does in n steps. Then, according to Lemma 1, the proof is complete. $\square$

The following theorem presents the main result of this section.

Theorem 2. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} = \mathscr{L}^n_R$.*

Proof. This immediately follows from the previous two lemmas. $\square$

Corollary 3. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} \subset \mathscr{L}^{n+1}_{\{1,2\}}$.*

Proof. This follows from the previous theorem and Theorem 2.3 in [10]. $\square$

3.4 Language Families $\mathscr{L}^n_{\{1,2,3,4\}}$

The following lemma shows that any language accepted by an nPPDA can be generated by a programmed grammar of index $n+1$.

Lemma 8. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2,3,4\}} \subseteq \mathscr{L}^{n+1}_P$.*

Before the formal proof of the lemma, we provide some explanations to the construction. Informally, to an nPPDA M, we construct a programmed grammar, G, of index $n+1$ so that the ith nonterminal of G, which is of the form $\langle A_i,k\rangle$, $1 \leq k \leq n+1$, is associated with the ith pushdown. Specifically, if the current content of M's pushdowns is $c_2c_1\$b_2b_1\$a_2a_1\$$ (corresponding to a string $a_1a_2b_1b_2c_1c_2$), then the sentential form of G is of the form $\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$. Then, the pop operation is simulated so that

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$$

is replaced with

$$a_1a_2\langle A_1,3\rangle b_1b_2\langle A_2,3\rangle c_1c_2\langle A_3,3\rangle\,.$$

The push operation pushing a onto the second pushdown, i.e., $c_2c_1\$b_2b_1a\$a_2a_1\$$ corresponding to a string $a_1a_2ab_1b_2c_1c_2$, is simulated by replacing

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$$

with

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle ab_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle\,.$$

The operation introducing a new, say the first, pushdown, i.e., $c_2c_1\$b_2b_1\$a_2a_1\$\$$, is simulated by replacing

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$$

with

$$\langle A_1,5\rangle\langle A_2,5\rangle a_1a_2\langle A_3,5\rangle b_1b_2\langle A_4,5\rangle c_1c_2\langle A_5,5\rangle\,.$$

Note that the previous first pushdown is the second from now on (till the other change). Finally, the join operation of the first and the second pushdown (by a state of the form $\langle r, join, 2\rangle$), i.e., $c_2c_1\$b_2b_1a_2a_1\$$, is simulated by replacing

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$$

with

$$\langle A_1,3\rangle a_1a_2b_1b_2\langle A_2,3\rangle c_1c_2\langle A_3,3\rangle\,.$$

The formal proof of Lemma 8 follows.

Proof. Let $M=(Q,T,R,s)$ be an nPPDA. Construct the following programmed grammar $G=(N,T,P,S)$, where $N=Q\times\{1,\dots,n+1\}$ and P is constructed as follows.

Set $f(r)=\{t : r\to t\in R\}$, and $g(f(r))=\bigcup_{p\in f(r)} g(p)$ (the definition of $g(p)$ follows).

1. For any rule $s\to p\in R$, add
   a) $(S\to\langle A_1,n+1\rangle\langle A_2,n+1\rangle\dots\langle A_{n+1},n+1\rangle, g(p))$ into P;
2. For all $p\in\mathscr{S}_1$ and $1\le l\le n+1$, add
   a) $([\mathbf{p},l,p]:\langle A_1,l\rangle\to\varepsilon,\{[/,2,l,p]:[/,2,l,p]\in lab(P)\})$;
3. For all $p\in\mathscr{S}_1\cup\mathscr{S}_4$ and $1\le i,l\le n+1$, $i\ge 2$, add
   a) $([/,i,l,p]:\langle A_i,l\rangle\to\langle A_{i-1},l\rangle,\{[/,i+1,l,p]\})$, for $i<l$;
   b) $([/,l,l,p]:\langle A_l,l\rangle\to\langle A_{l-1},l\rangle,\{[-,1,l,p]\})$;
4. For all $p\in\mathscr{S}_1\cup\mathscr{S}_4$ and $1\le i,l\le n+1$, $l\ge 2$, add
   a) $([-,i,l,p]:\langle A_i,l\rangle\to\langle A_i,l-1\rangle,\{[-,i+1,l,p]\})$, for $i<l-1$;
   b) $([-,l-1,l,p]:\langle A_{l-1},l\rangle\to\langle A_{l-1},l-1\rangle, g(f(p)))$;
5. For all $p\in\mathscr{S}_2$ and $1\le i,l\le n+1$, add
   a) $([i,l,p]:\langle A_i,l\rangle\to\langle A_i,l\rangle a, g(f(p)))$;
6. For all $p\in\mathscr{S}_3$ and $1\le i,l\le n+1$, $i\le n$, add
   a) $([*,i,l,i,p]:\langle A_i,l\rangle\to\langle A_{i+1},l\rangle,\{[*,i+1,l,i,p]\})$, for $i<l$;
   b) $([*,l,l,i,p]:\langle A_l,l\rangle\to\langle A_{l+1},l\rangle,\{[\mathbf{n},i+1,l,p]\})$;

7. For all $p \in \mathscr{S}_3$, $1 \leq l \leq n+1$ and $1 < i \leq n+1$, add
   a) $([\mathbf{n}, i, l, p] : \langle A_i, l\rangle \to \langle A_{i-1}, l\rangle\langle A_i, l\rangle, \{[+, 1, l, p]\})$;
8. For all $p \in \mathscr{S}_3$ and $1 \leq i, l < n+1$, add
   a) $([+, i, l, p] : \langle A_i, l\rangle \to \langle A_i, l+1\rangle, \{[+, i+1, l, p]\})$, for $i < l+1$;
   b) $([+, l+1, l, p] : \langle A_{l+1}, l\rangle \to \langle A_{l+1}, l+1\rangle, g(f(p)))$;
9. For all $p \in \mathscr{S}_4$ and $1 \leq i, l \leq n+1$, add
   a) $([\mathbf{j}, i, l, p] : \langle A_i, l\rangle \to \varepsilon, W)$, $W = \{[/, i+1, l, p]\})$ if $i < l$, $W = \{[-, 1, l, p]\}$ otherwise.

$g(p)$ depends on p as follows:

$p = \langle r, \boldsymbol{pop}\rangle$**:** $g(p) = \{[\mathbf{p}, l, p] : [\mathbf{p}, l, p] \in lab(P)\}$;

$p = \langle r, \boldsymbol{push}, i, a\rangle$**:** $g(p) = \{[i, l, p] : [i, l, p] \in lab(P)\}$;

$p = \langle r, \boldsymbol{new}, i\rangle$**:** $g(p) = \{[*, i, l, i, p] : [*, i, l, i, p] \in lab(P)\}$;

$p = \langle r, \boldsymbol{join}, i\rangle$**:** $g(p) = \{[\mathbf{j}, i, l, p] : [\mathbf{j}, i, l, p] \in lab(P)\}$.

Consider a configuration $w_k\$\ldots\$w_2\$w_1\$pw$ of M and the corresponding sentential form of G, i.e., $(\langle A_1, k+1\rangle w_1\langle A_2, k+1\rangle w_2\ldots\langle A_k, k+1\rangle w_k\langle A_{k+1}, k+1\rangle, g(p))$. If $p = \langle r, pop\rangle$, G simulates the computational step as follows:

$$\begin{array}{ll} & (\langle A_1,k+1\rangle w_1\langle A_2,k+1\rangle w_2\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, \{[\mathbf{p},k+1,p]\}) \\ \Rightarrow & (w_1\langle A_2,k+1\rangle w_2\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, \{[/,2,k+1,p]\}) \\ \Rightarrow^k & (w_1\langle A_1,k+1\rangle w_2\ldots\langle A_{k-1},k+1\rangle w_k\langle A_k,k+1\rangle, \{[-,1,k+1,p]\}) \\ \Rightarrow^k & (w_1\langle A_1,k\rangle w_2\ldots\langle A_{k-1},k\rangle w_k\langle A_k,k\rangle, g(f(p))). \end{array}$$

If $p = \langle r, push, i, a\rangle$, G simulates the computational step as follows:

$$\begin{array}{ll} & (\langle A_1,k+1\rangle w_1\ldots\langle A_i,k+1\rangle w_i\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, \{[i,k+1,p]\}) \\ \Rightarrow & (\langle A_1,k+1\rangle w_1\ldots\langle A_i,k+1\rangle aw_i\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, g(f(p))). \end{array}$$

If $p = \langle r, new, i\rangle$, G simulates the computational step as follows:

$$\begin{array}{ll} & (\langle A_1,k+1\rangle w_1\ldots w_{i-1}\langle A_i,k+1\rangle w_i\ldots w_k\langle A_{k+1},k+1\rangle, \{[*,i,k+1,i,p]\}) \\ \Rightarrow^{k-i+1} & (\ldots w_{i-1}\langle A_{i+1},k+1\rangle w_i\ldots w_k\langle A_{k+2},k+1\rangle, \{[\mathbf{n},i+1,k+1,p]\}) \\ \Rightarrow & (\ldots w_{i-1}\langle A_i,k+1\rangle\langle A_{i+1},k+1\rangle w_i\ldots w_k\langle A_{k+2},k+1\rangle, \{[+,1,k+1,p]\}) \\ \Rightarrow^{k+2} & (\langle A_1,k+2\rangle w_1\ldots w_k\langle A_{k+2},k+2\rangle, g(f(p))). \end{array}$$

If $p = \langle r, join, i\rangle$, G simulates the computational step as follows:

$$\begin{array}{ll} & (\langle A_1,k+1\rangle w_1\ldots w_{i-1}\langle A_i,k+1\rangle w_i\ldots w_k\langle A_{k+1},k+1\rangle, \{[\mathbf{j},i,k+1,p]\}) \\ \Rightarrow & (\ldots\langle A_{i-1},k+1\rangle w_{i-1}w_i\langle A_{i+1},k+1\rangle w_{i+1}\ldots, \{[/,i+1,k+1,p]\}) \\ \Rightarrow^{k-i} & (\ldots\langle A_{i-1},k+1\rangle w_{i-1}w_i\langle A_i,k+1\rangle w_{i+1}\ldots w_k\langle A_k,k+1\rangle, \{[-,1,k+1,p]\}) \\ \Rightarrow^k & (\langle A_1,k\rangle w_1\ldots\langle A_{i-1},k\rangle w_{i-1}w_i\langle A_i,k\rangle\ldots w_k\langle A_k,k\rangle, g(f(p))). \end{array}$$

As any derivation of G simulates a computation of M, we have $L(M) = L(G)$. $\square$

The next lemma shows that any language generated by a programmed grammar of index n is accepted by an $(n+1)$PPDA.

Lemma 9. *For all* $n \geq 1$, $\mathscr{L}_P^n \subseteq \mathscr{L}_{\{1,2,3,4\}}^{n+1}$.

The main idea of the proof is to simulate a derivation of a programmed grammar, G, of index n by a generalized $(n+1)$PPDA, M, so that what G generates to the right of the rewritten nonterminal, say $Aw_1Bw_2Cw_3 \Rightarrow Aw_1B'uw_2Cw_3$, M pushes to its corresponding pushdown, $w_3^R\$w_2^Ru^R\$w_1^R\$$. If G generates a string, v, to the left of the rewritten nonterminal, say $Aw_1Bw_2Cw_3 \Rightarrow Aw_1vB''w_2Cw_3$, then M creates a new pushdown just before the pushdown corresponding to the rewritten nonterminal, $w_3^R\$w_2^R\$\$w_1^R\$$, pushes v^R to the new pushdown, $w_3^R\$w_2^R\$v^R\$w_1^R\$$, and joins the two pushdowns, $w_3^R\$w_2^R\$v^Rw_1^R\$$. By this, M puts v^R to the bottom of the pushdown. In case of the first pushdown, the join operation is replaced with the pop operation. The formal proof follows.

Proof. Let $G = (N,T,P,S)$ be a programmed grammar of index n, for some $n \geq 1$. Construct a generalized $(n+1)$PPDA $M = (Q,T,R,s)$ as follows.

1. Set $Q = (lab(P) \cup \{+\}) \times \bigcup_{k \leq n} N^k \times \{0,1,\dots,m+1\}$, for $m = \max\{k : A \to u \in P, occur(u,N) = k\}$;

2. For all $p : A \to u_1B_1u_2B_2\dots u_kB_ku_{k+1} \in P$, where $u_i \in T^*$ and $B_j \in N$, for all $i = 1,\dots,k+1$, $j = 1,\dots,k$, $k \geq 0$, and for all $\langle +, \alpha A\beta, 0\rangle \in Q$, where $\alpha,\beta \in N^*$, and $l = occur(\alpha A, N)$, add the following to R:

   - $s \to \langle\langle +,S,0\rangle, push, 1, \varepsilon\rangle$,
   - $\langle\langle +,\alpha A\beta,0\rangle, push, 1, \varepsilon\rangle \to \langle\langle p,\alpha B_1\dots B_k\beta,k+1\rangle, push, l+k-1, u_{k+1}^R\rangle$,
   - $\langle\langle p,\alpha B_1\dots B_k\beta,k+1\rangle, push, l+k-1, u_{k+1}^R\rangle \to$ $\langle\langle p,\alpha B_1\dots B_k\beta,k\rangle, push, l+k-2, u_k^R\rangle$,
   - $\langle\langle p,\alpha B_1\dots B_k\beta,k\rangle, push, l+k-2, u_k^R\rangle \to$ $\langle\langle p,\alpha B_1\dots B_k\beta,k-1\rangle, push, l+k-3, u_{k-1}^R\rangle$,

     $\vdots$

   - $\langle\langle p,\alpha B_1\dots B_k\beta,2\rangle, push, l, u_2^R\rangle \to \langle\langle p,\alpha B_1\dots B_k\beta,1\rangle, new, l\rangle$,
   - $\langle\langle p,\alpha B_1\dots B_k\beta,1\rangle, new, l\rangle \to \langle\langle p,\alpha B_1\dots B_k\beta,1\rangle, push, l, u_1^R\rangle$,
   - if $l = 1$, add
     - $\langle\langle p,B_1\dots B_k\beta,1\rangle, push, 1, u_1^R\rangle \to \langle\langle p,B_1\dots B_k\beta,0\rangle, pop\rangle$,
     - $\langle\langle p,B_1\dots B_k\beta,0\rangle, pop\rangle \to \langle\langle +,B_1\dots B_k\beta,0\rangle, push, 1, \varepsilon\rangle$,
   - if $l \geq 2$, add
     - $\langle\langle p,\alpha B_1\dots B_k\beta,1\rangle, push, l, u_1^R\rangle \to \langle\langle p,\alpha B_1\dots B_k\beta,0\rangle, join, l\rangle$,
     - $\langle\langle p,\alpha B_1\dots B_k\beta,0\rangle, join, l\rangle \to \langle\langle +,\alpha B_1\dots B_k\beta,0\rangle, push, 1, \varepsilon\rangle$.

We have proved that $L(M) = L(G)$, where M is a generalized $(n+1)PPDA$. The proof now follows by Lemma 1. □

Let $n \geq 1$. Analogously as in [2, Theorem 3.1.7], we can prove that the language

$$L_n = \{b(a^i b)^{2n-1} : i \geq 1\} \in \mathscr{L}_P^n - \mathscr{L}_P^{n-1}\,.$$

Lemma 10. *For all $n \geq 1$, $L_n \in \mathscr{L}^n_{\{1,2,3,4\}}$.*

Informally, the automaton has n pushdowns and each but the one of them contains $a^i ba^i$, for some $i \geq 1$. Thus, two symbols a are put to a pushdown – one to the top and one to the bottom. Finally, the symbol b is pushed to the bottom of all $n-1$ pushdowns, i.e., they contain the string $a^i ba^i b$. Obviously, by the operations new and pop, $ba^i b$ can be simulated and compared with the prefix of the input symbol by symbol during the computation. Thus, the automaton has read $ba^i b$, and the content of each of $n-1$ pushdowns is $a^i ba^i b$, i.e., the automaton has accepted the string $ba^i b(a^i b)^{2(n-1)} = b(a^i b)^{2n-1}$.

Proof. If $n = 1$, the proof is trivial; just push $ba^i b$ to the pushdown. Thus, let $n \geq 2$ and $M = (Q, \{a,b\}, R, s)$ be an nPPDA, where $Q = \{0, p, q, r, s, t, f\}$, and R is constructed as follows.

Phase 1.

1. $s \to \langle 0, push, 1, b\rangle$,
2. $\langle 0, push, 1, b\rangle \to \langle 0, pop\rangle$,
3. $\langle 0, pop\rangle \to \langle p, push, 1, b\rangle$,
4. for $2 \leq i < n-1$,

4a. $\langle p, push, i, b\rangle \to \langle p, push, i+1, b\rangle$,

4b. $\langle p, push, n-1, b\rangle \to \langle q, new, 1\rangle$,

Phase 2.

5. $\langle q, new, 1\rangle \to \langle q, push, 1, a\rangle$,
6. $\langle q, push, 1, a\rangle \to \langle q, pop\rangle$,
7. $\langle q, pop\rangle \to \langle s, push, 1, a\rangle$,
8. for $1 \leq i < n$,

8a. $\langle s, push, i, a\rangle \to \langle r, new, i+1\rangle$,

8b. $\langle r, new, i+1\rangle \to \langle r, push, i+1, a\rangle$,

8c. $\langle r, push, i+1, a\rangle \to \langle r, join, i+1\rangle$,

8d. $\langle r, join, i\rangle \to \langle s, push, i, a\rangle$, $i \geq 2$,

8e. $\langle r, join, n\rangle \to \langle q, new, 1\rangle$,

8f. $\langle r, join, n\rangle \to \langle t, new, 1\rangle$,

Phase 3.

9. $\langle t, new, 1\rangle \to \langle t, push, 1, b\rangle$,
10. $\langle t, push, 1, b\rangle \to \langle t, pop\rangle$,
11. $\langle t, pop\rangle \to \langle t, new, 2\rangle$,
12. for $2 \leq i \leq n$,

12a. $\langle t, new, i\rangle \to \langle t, push, i, b\rangle$,

12b. $\langle t, push, i, b\rangle \to \langle t, join, i\rangle$,

12c. $\langle t, join, i\rangle \to \langle t, new, i+1\rangle$,

Phase 4.

13. $\langle t, new, n+1\rangle \to \langle f, pop\rangle$,
14. $\langle f, pop\rangle \to \langle f, pop\rangle$.

Phase 1 reads b from the input and pushes b to $n-1$ pushdowns. Phase 2 repeatedly reads a from the input and pushes a on the top and to the bottom of all $n-1$ pushdowns. Phase 3 reads b from the input and pushes b to the bottom of all $n-1$ pushdowns. Finally, Phase 4 pops all $n-1$ pushdowns. Clearly, ba^ib has been read from the input and each of $n-1$ pushdowns contains $ba^iba^i\$$, where the top of the pushdown is on the right. Thus, we have $L(M) = L_n$. □

Corollary 4. *For all $n \geq 1$, $\mathscr{L}_P^n \subset \mathscr{L}_{\{1,2,3,4\}}^{n+1}$.*

Proof. The inclusion follows from Lemma 9 and the strictness from Lemma 10. □

The following corollary summarizes the power of nPPDAs known so far.

Corollary 5. *For all $n \geq 1$, $\mathscr{L}_{\{1,2,3,4\}}^n \subseteq \mathscr{L}_P^{n+1} \subset \mathscr{L}_{\{1,2,3,4\}}^{n+2}$.*

Proof. It follows immediately from Lemmas 8 and 9, and the previous corollary. □

Analogously, we can prove that for all $n \geq 2$,

$$K_{n+1} = \{a_1^k a_2^k \dots a_{n+1}^k : k \geq 1\} \in \mathscr{L}_{\{1,2,3,4\}}^n \,,$$

which proves the following result.

Corollary 6. *For all $n \geq 2$, $\mathscr{L}_{\{1,2\}}^n \subset \mathscr{L}_{\{1,2,3,4\}}^n$.*

Proof. Ibarra [10, Theorem 2.3] proved that $K_{n+1} \notin \mathscr{L}_R^n = \mathscr{L}_{\{1,2\}}^n$. □

Note that by the trick pushing the content of one pushdown to the bottom of the other, we can prove that for all $n \geq 1$, $K_{2n-1} \in \mathscr{L}_{\{1,2,3,4\}}^n$.

4 Conclusion

In this paper, we discussed two variants of pure multi-pushdown automata that perform complete pushdown pops and proved two infinite language hierarchies they characterize with respect to the number of pushdowns. The following theorem summarizes the results of this paper.

Theorem 3. *1. $\mathscr{L}_{REG} = \mathscr{L}_{\{1,2\}}^1 = \mathscr{L}_{\{1,2,4\}}^1 = \mathscr{L}_{\{1,2,3\}}^1 = \mathscr{L}_{\{1,2,3,4\}}^1$.*

2. For all $n \geq 2$, $\mathscr{L}_{\{1,2\}}^n = \mathscr{L}_{\{1,2,4\}}^n \subset \mathscr{L}_{\{1,2,3\}}^n \subseteq \mathscr{L}_{\{1,2,3,4\}}^n$.

3. For all $n \geq 1$, $\mathscr{L}_{\{1,2\}}^n \subset \mathscr{L}_{\{1,2\}}^{n+1}$.

4. For all $n \geq 1$, $\mathscr{L}_{\{1,2,3,4\}}^n \subset \mathscr{L}_{\{1,2,3,4\}}^{n+2}$.

Moreover, note that by Corollary 3.4 in [3], Corollary of Lemma 3.1.5 in [2], and Corollary 5, any language $L \in \bigcup_{n=1}^{\infty} \mathscr{L}^n_{\{1,2,3,4\}}$ over a one-letter alphabet is regular.

On the other hand, this paper does not answer the question of whether the inclusions $\mathscr{L}^n_{\{1,2,3,4\}} \subseteq \mathscr{L}^{n+1}_{\{1,2,3,4\}}$, $n \geq 1$, and $\mathscr{L}^n_{\{1,2,3\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$, $n \geq 2$, are proper or not. However, we conjecture that these inclusions are proper. Furthermore, one of the interesting questions concerning this is whether the language

$$M_n = \{w^n : w \in \{a,b\}^*\}$$

is in $\mathscr{L}^{n-1}_{\{1,2,3,4\}}$, for $n \geq 2$. This is of interest because if M_n is not in $\mathscr{L}^{n-1}_{\{1,2,3,4\}}$, then it implies that

1. $M_n \in \mathscr{L}^n_{\{1,2\}} \cap (\mathscr{L}^n_{\{1,2,3,4\}} - \mathscr{L}^{n-1}_{\{1,2,3,4\}})$ and that
2. $\mathscr{L}^n_{\{1,2\}} \not\subseteq \mathscr{L}^{n-1}_{\{1,2,3,4\}}$,

as it is not hard to see that $M_n \in \mathscr{L}^n_{\{1,2\}}$. Another interesting question is whether the language $K_{n+1} \in \mathscr{L}^n_{\{1,2,3\}}$ because if this is not true, then it implies $\mathscr{L}^n_{\{1,2,3\}} \subset \mathscr{L}^n_{\{1,2,3,4\}}$, for $n \geq 2$. Finally, the following question is of interest from the viewpoint of descriptional complexity: what is the power of pure multi-pushdown automata that perform complete pushdown pops with respect to the number of states?

Acknowledgements

The authors gratefully acknowledge useful suggestions and comments of the anonymous referees.

This work was supported by the Czech Ministry of Education under the Research Plan No. MSM 0021630528 and the Czech Grant Agency project No. GA201/07/0005.

References

[1] Courcelle, B. On jump deterministic pushdown automata. *Math. Systems Theory*, 11:87–109, 1977.

[2] Dassow, J. and Păun, Gh. *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[3] Fernau, H. and Holzer, M. Regulated finite index language families collapse. Technical report, University of Tuebingen, 1996.

[4] Fischer, P. C. Multi-tape and infinite-state automata—a survey. *Commun. ACM*, 8(12):799–805, 1965.

[5] Fischer, P. C. and Rosenberg, A. L. Multitape one-way nonwriting automata. *J. Comput. System Sci.*, 2:88–101, 1968.

[6] Ginsburg, S., Greibach, S. A., and Harrison, M. A. One-way stack automata. *J. ACM*, 14:389–418, 1967.

[7] Ginsburg, S. and Spanier, E. Finite-turn pushdown automata. *SIAM J. Control*, 4:429–453, 1968.

[8] Greibach, S. A. Checking automata and one-way stack languages. *J. Comput. System Sci.*, 3:196–217, 1969.

[9] Hopcroft, J. E. and Ullman, J. D. *Formal Languages and Their Relation to Automata*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.

[10] Ibarra, O. H. Simple matrix languages. *Inform. and Control*, 17(4):359–394, 1970.

[11] Meduna, A. Simultaneously one-turn two-pushdown automata. *Int. J. Comp. Math.*, 80:679–687, 2003.

[12] Meduna, A. Deep pushdown automata. *Acta Inform.*, 42(8–9):541–552, 2006.

[13] Meduna, A. and Masopust, T. Self-regulating finite automata. *Acta Cybernet.*, 18:135–153, 2007.

[14] Sakarovitch, J. Pushdown automata with terminating languages. *Languages and Automata Symposium, RIMS 421, Kyoto University*, pages 15–29, 1981.

[15] Salomaa, A. *Formal languages*. Academic Press, New York, 1973.

[16] Siromoney, R. On equal matrix languages. *Inform. and Control*, 14:135–151, 1969.

[17] Siromoney, R. Finite-turn checking automata. *J. Comput. System Sci.*, 5:549–559, 1971.

[18] Valiant, L. The equivalence problem for deterministic finite turn pushdown automata. *Inform. and Control*, 81:265–279, 1989.

[19] Wood, D. *m*-parallel *n*-right linear simple matrix languages. *Util. Math.*, 8:3–28, 1975.

Received ...