Uniform Distributed Pushdown Automata Systems

Fernando Arroyo , Juan Castellanos , and Victor Mitrana

Abstract. A distributed pushdown automata system consists of several pushdown automata which work in turn on the input word placed on a common one-way input tape under protocols and strategies similar to those in which cooperating distributed (CD) grammar systems work. Unlike the CD grammar systems case, where one may add or remove duplicate components without modifying the generated language, the identical components play an important role in distributed pushdown automata systems. We consider here uniform distributed pushdown automata systems (UDPAS), namely distributed pushdown automata systems having all components identical pushdown automata.

We consider here just a single protocol for activating/deactivating components, namely a component stays active as long as it can perform moves, as well as two ways of accepting the input word: by empty stacks (all components have empty stacks) or by final states (all components are in final states), when the input word is completely read. We mainly investigate the computational power of UDPAS accepting by empty stacks and a few decidability and closure properties of the families of languages they define. Some directions for further work and open problems are also discussed.

1 Introduction

In the last decades, researchers and practitioners have shown an increasing interest in distributed systems. Among various models a formal language theoretic paradigm called *grammar system* has been proposed [5]. Two main architectures

* Work supported by the Project TIN2011-28260-C03-03.

have been distinguished in the area, cooperating distributed (CD) grammar systems [4] and parallel communicating (PC) grammar systems [15]. Several motivations have been involved in introducing the CD grammar system concept:

– A generalization of the two-level substitution grammars. This was the main purpose of the paper [14] where the syntagma *cooperating grammar system* was proposed.
– In the architecture of a CD grammar system one can recognize the structure of a blackboard model, as used in problem-solving area: the common sentential form is the "blackboard" (the common data structure containing the current state of the problem which is to be solved), the component grammars are the knowledge sources contributing to solving the problem, the protocol of cooperation encodes the control on the work of the knowledge sources [10]. This was the explicit motivation of [4], the paper where CD grammar systems were introduced in the form we consider here.
– The increase of the computational power of components by cooperation and communication and the decrease of the complexity of different tasks by distribution and parallelism.

An entire theory has been developed for both types of grammar systems, see the monograph [5] and more recently the chapter [8] in [16]. The obtained results showed that cooperation and communication increases the power of individual components: large language classes were described by systems of very simple grammars belonging to grammar classes with weak computational power. In spite of this notable development, very little has been done with respect to automata systems working under similar strategies. These investigations might be of particular interest from several points of view: they might lead to a comparison between the power of distributed generative and accepting devices, and might give information on the boundaries of describing language classes in terms of automata systems. A series of papers [6,12,13,3,2] was devoted to PC automata systems whose components are finite or pushdown automata. We briefly recall the previous works dealing with distributed systems formed by automata done in this respect.

In [9] some special types of multi-stack pushdown automata were introduced. These mechanisms are usual multi-stack pushdown automata whose stacks cooperate in the accepting process under some strategies borrowed from CD grammar systems. However, they cannot be seen as the automata counterpart of CD grammar systems. A similar approach has been reported in [11].

The first (and unique so far) work considering systems of pushdown automata whose working mode is very close to that of CD grammar systems is [7]. A distributed pushdown automata system (DPAS) has a common one-way input tape, one reading head, and several central units. Each central unit is in a state from its own finite sets of states and accesses the topmost of its own pushdown memory. At any moment only one central unit is active, the others are "frozen". When active, the central unit can also read the current input symbol by means of the common reading head. Activation of some component means that the central

unit of that component takes control over the reading head. We defined several protocols for activating components. Two ways of accepting were defined: by empty stacks or by final states meaning that all components have empty stacks or are in final states, respectively, when the input word is completely read.

This note considers a problem of interest in our view which is represented by the DPAS with identical components, that is all components are identical pushdown automata. Such DPAS are called here *uniform DPAS (UDPAS)*. This aspect makes no difference for CD grammar systems; in other words, one can add or remove identical components in a CD grammar system without modifying the generated language. Unlike the CD grammar systems case, the identical components play an important role in DPAS as we shall see in the sequel. Returning to the original motivation mentioned in the beginning of this paper (blackboard model of problem solving), it is not artificial to assume that all agents which participate in the problem solving process have the same knowledge. This approach suggests a close connection with *amorphous systems*: (i) each component has rather modest computing power, (ii) each component is programmed identically though each has means for storing local state and memory, (iii) each component has no a priori knowledge of its position within the system.

We first prove that UDPAS accepting by final states are strictly more powerful than UDPAS accepting by empty stacks. Then we mainly consider UDPAS accepting by empty stacks and investigate their computational power and a few decidability and closure properties of the families of languages they define. Some directions for further work and open problems are also discussed.

2 Basic Definitions

We assume the reader to be familiar with the basic concepts in automata and formal language theory; for further details, we refer to [16].

An alphabet is a finite and nonempty set of symbols. Any sequence of symbols from an alphabet V is called word over V. For an alphabet V, we denote by V^* the free monoid generated by V under the operation of concatenation; the empty word is denoted by ε and the semigroup $V^* - \{\varepsilon\}$ is denoted by V^+. The length of $x \in V^*$ is denoted by $|x|$ while $|x|_a$ denotes the number of occurrences of the symbol a in x. A subset of V^* is called language over V. The Parikh mapping over an alphabet $V = \{a_1, a_2, \ldots, a_k\}$ denoted by ψ_V is a morphism from V^* to $\mathbb{N}^k$, where $\psi_V(a_i)$ is the vector having all its entries equal to 0 except the i-th entry which is 1. If $L \subseteq V^*$, then the Parikh image of L is $\psi(L) = \{\psi_V(x) \mid x \in L\}$. We omit the subscript V whenever the alphabet is understood from the context.

We shall also denote by $Rec_X(A)$ the language accepted by a pushdown automaton A with final state if $X = f$, or with empty stack if $X = \varepsilon$. We note that pushdown automata characterize the class of context-free languages in both modes of acceptance. The family of context-free languages is denoted by CF. Remember that the Parikh image of any context-free language is a semilinear set.

We now give the definition of the main concept of the paper following [7]. A *distributed pushdown automata system* (DPAS for short) of degree n is a construct

$$\mathcal{A} = (V, A_1, A_2, \ldots, A_n),$$

where V is an alphabet and for each $1 \leq i \leq n$, $A_i = (Q_i, V, \Gamma_i, f_i, q_i, Z_i, F_i)$ is a nondeterministic pushdown automaton with the set of states Q_i, the initial state $q_i \in Q_i$, the alphabet of input symbols V, the alphabet of pushdown symbols Γ_i, the initial contents of the pushdown memory $Z_i \in \Gamma_i$, the set of final states $F_i \subseteq Q_i$, and the transition mapping f_i from $Q_i \times V \cup \{\varepsilon\} \times \Gamma_i$ into the finite subsets of $Q_i \times \Gamma_i^*$. We refer to the automaton A_i, $1 \leq i \leq n$, as the i^{th} component of $\mathcal{A}$.

An *instantaneous description* (ID) of a DPAS as above is $2n+1$-tuple

$$(x, s_1, \alpha_1, s_2, \alpha_2, \ldots, s_n, \alpha_n),$$

where $x \in V^*$ is the part of the input word to be read, and for each $1 \leq i \leq n$, s_i is the current state of the automaton A_i and $\alpha_i \in \Gamma_i^*$ is the pushdown memory content of the same automaton.

A one step move of $\mathcal{A}$ done by the component i, $1 \leq i \leq n$, is represented by a binary relation $\vdash_i$ on all IDs defined in the following way:

$$(ax, s_1, \alpha_1, s_2, \alpha_2, \ldots, s_i, \alpha_i, \ldots, s_n, \alpha_n) \vdash_i (x, s_1, \alpha_1, s_2, \alpha_2, \ldots, r_i, \beta, \ldots, s_n, \alpha_n)$$

if and only if $(r_i, \delta) \in f_i(s_i, a, A)$, where $a \in V \cup \{\varepsilon\}$, $\alpha_i = A\gamma$, and $\beta = \delta\gamma$.

As usual, $\vdash_i^*$ denotes the reflexive and transitive closure of $\vdash_i$. Let now C_1, C_2 be two IDs of a DPAS. We say that C_1 directly derives C_2 by a move representing a sequence of steps done by the component i that cannot be continued, denoted by $C_1 \vdash_{\mathcal{A}}^t C_2$, if and only if $C_1 \vdash_i^* C_2$ for some $1 \leq i \leq n$, and there is no C' with $C_2 \vdash_i C'$. In other words, as soon as a component is activated, it remains active as long as it is possible.

The language accepted by a DPAS $\mathcal{A}$ as above by final states is defined by

$$\begin{aligned} Rec_f(\mathcal{A}) = \{w \mid w \in V^*,\ & (w, q_1, Z_1, q_2, Z_2, \ldots, q_n, Z_n)(\vdash_{\mathcal{A}}^t)^* \\ & (\varepsilon, s_1, \alpha_1, s_2, \alpha_2, \ldots, s_n, \alpha_n) \text{ with } \alpha_i \in \Gamma_i^*,\ s_i \in F_i,\ \text{for all } 1 \leq i \leq n\} \end{aligned}$$

Similarly, the language accepted by DPAS $\mathcal{A}$ as above by empty stacks is defined by

$$\begin{aligned} Rec_\varepsilon(\mathcal{A}) = \{w \mid w \in V^*,\ & (w, q_1, Z_1, q_2, Z_2, \ldots, q_n, Z_n)(\vdash_{\mathcal{A}}^t)^* \\ & (\varepsilon, s_1, \varepsilon, s_2, \varepsilon, \ldots, s_n, \varepsilon) \text{ for some } s_i \in Q_i, 1 \leq i \leq n\} \end{aligned}$$

For the rest of this paper we consider *uniform DPAS* (UDPAS) only. A DPAS $\mathcal{A} = (V, A_1, A_2, \ldots, A_n)$ with $A_1 = A_2 = \cdots = A_n = A$, which is simply denoted by $\mathcal{A} = (n, V, A)$, is said to be uniform. Therefore, for each UDPAS it suffices to give its degree (number of components) and the pushdown automaton. We illustrate the above notions through an example which will also be useful in the sequel.

Example 1. *Let $\mathcal{A}$ be the UDPAS of degree 2 with the pushdown automaton defined by the following transition mapping:*

$f(q_0, X, Z_0) = \{(s_X, XZ_0)\}, X \in \{a, b\}$ $\quad f(s_X, X, X) = \{(s_X, XX)\}, X \in \{a, b\},$
$f(s_a, c, a) = \{(s_a, \varepsilon)\}$ $\quad f(s_b, d, b) = \{(s_b, \varepsilon)\},$
$f(s_X, X, Z_0) = \{(s_X, XZ_0)\}, X \in \{a, b\}$ $\quad f(s_X, \varepsilon, Z_0) = \{(s, \varepsilon)\}, X \in \{a, b\}$

The set $\{s\}$ is the set of final states. We first note that the language accepted by the pushdown automaton by final states/empty stack is $L_0 = D_{a,c} \cup D_{b,d}$. Here $D_{x,y}$ is the Dyck language over the alphabet $\{x, y\}$. Second, the language recognized by $\mathcal{A}$ by final states/empty stacks is the language L_1 that includes $D^2_{a,c} \cup D^2_{b,d}$ and all words formed by interleaving words from $D_{a,c}$ and $D_{b,d}$.

The families of languages accepted by UDPAS of degree n by final states or empty stacks are denoted by $\mathcal{L}_f(UDPAS, n)$ or $\mathcal{L}_\varepsilon(UDPAS, n)$, respectively.

Example 1 shows a strong connection between the languages recognized by (U)DPAS and languages obtained by means of the following operation intensively investigated in the formal language and concurrency theory. The *shuffle* operation applied to two words leads to the set of all words obtained from the original two words by interleaving their letters but keeping their order in the two words like interleaving two decks of cards. Formally, this operation is defined recursively on words over an alphabet V as follows:

$$\text{ш}\,(\varepsilon, x) = \text{ш}\,(x, \varepsilon) = \{x\}, \text{ for any } x \in V^*$$
$$\text{ш}\,(ax, by) = \{a\}\,\text{ш}\,(x, by) \cup \{b\}\,\text{ш}\,(ax, y), \text{ for all } a, b \in V,\ x, y \in V^*.$$

This operation may naturally be extended to languages and to have k arguments as

$$\text{ш}\,(L_1, L_2) = \bigcup_{x \in L_1, y \in L_2} \text{ш}\,(x, y),$$

and $\text{ш}_k(x_1, x_2, \ldots, x_k) = \text{ш}\,(\,\text{ш}_{k-1}(x_1, x_2, \ldots, x_{k-1}), \{x_k\})$, respectively. Also, ш_k is extended to languages as

$$\text{ш}_k(L_1, L_2, \ldots, L_k) = \bigcup_{x_i \in L_i, 1 \le i \le k} \text{ш}_k(x_1, x_2, \ldots, x_k).$$

If each language $L_1, L_2, \ldots, L_k$ equals L, we denote

$$\text{ш}^0(L) = \{\varepsilon\},$$
$$\text{ш}^{k+1}(L) = \text{ш}\,(\,\text{ш}^k(L), L)), \text{ for all } k \ge 0,$$
$$\text{ш}^*(L) = \bigcup_{k \ge 0} \text{ш}^k(L).$$

3 Computational Power

It is worth mentioning in the beginning of this section that any context-free language can be accepted by a DPAS of degree n for all $n \ge 1$. This is not true anymore for UDPAS as we shall see in the sequel. We start with a result that will be useful in what follows.

Lemma 1. *For any UDPAS $\mathcal{A}$ of degree n there exists a context-free language L such that $L^n \subseteq Rec_\varepsilon(\mathcal{A}) \subseteq \text{ш}^n(L)$.*

Proof. Let $\mathcal{A}$ be a UDPAS formed by n copies of a pushdown automaton A. The statement follows immediately as soon as we take the context-free language as the language accepted with empty stack by A. □

It is known that pushdown automata accepting by empty stack or final state define the same class of languages. The situation is different for UDPAS.

Theorem 1. *$\mathcal{L}_\varepsilon(UDPAS, p) \subset \mathcal{L}_f(UDPAS, p)$, for all $p \geq 2$.*

Proof. The inclusion is proved by the standard construction that transforms a pushdown automaton accepting with empty stack into a pushdown automaton accepting with final states. For proving the properness of this inclusion we construct the UDPAS with $p \geq 2$ identical copies of the pushdown automaton A defined as follows:

$f(q_0, a, Z_0) = \{(s_1, aZ_0)\}$,
$f(s_1, a, a) = \{(s_1, aa)\}$
$f(s_1, X, a) = \{(s_e, a)\}, X \in \{b, c, d\}$
$f(s_1, \varepsilon, a) = \{(p_1, a)\}$
$f(p_1, c, a) = \{(p_1, \varepsilon)\}$
$f(p_1, X, a) = \{(s_e, a)\}, X \in \{a, b, d\}$
$f(p_1, \varepsilon, Z_0) = \{(s_f^1, Z_0)\}$
$f(s_f^1, X, Z_0) = \{(s_e, Z_0)\}, X \in \{a, b, c\}$
$f(q_0, \varepsilon, Z_0) = \{(s_f^3, Z_0)\}$.

$f(q_0, b, Z_0) = \{(s_2, bZ_0)\}$,
$f(s_2, b, b) = \{(s_2, bb)\}$,
$f(s_2, X, b) = \{(s_e, b)\}, X \in \{a, c, d\}$,
$f(s_2, \varepsilon, b) = \{(p_2, b)\}$,
$f(p_2, d, b) = \{(p_2, \varepsilon)\}$,
$f(p_2, X, b) = \{(s_e, b)\}, X \in \{a, b, c\}$,
$f(p_2, \varepsilon, Z_0) = \{(s_f^2, Z_0)\}$,
$f(s_f^2, X, Z_0) = \{(s_e, Z_0)\}, X \in \{a, b, c, d\}$,

It is easy to note that $Rec_f(A) = \{a^n c^n \mid n \geq 0\} \cup \{b^n d^n \mid n \geq 0\}$, where the set of final states of A is $\{s_f^1, s_f^2, s_f^3\}$. We now make a discussion about the language accepted by $\mathcal{A}$ with final states. First, any non-empty input word must start with either a or b. If the prefix of the input word composed by a is followed by a d, it is plain that the input word cannot be accepted. We now consider the case when the prefix of the input word composed by a is followed by c. Two situations may appear:

- The whole prefix of a's is processed continuously by the same component. In this case this component may either reach the final state s_f^1, if the input word is of the form $a^n c^n$ for some $n \geq 1$, or get stuck.
- Only a part of the prefix of a's is read by the first activated component; it follows that other components have to read the remaining part of this prefix. Now the next segment formed by c's of the input word will block at least one of all these two components.

Therefore, an input word of the form $a^+c^+(a+b+c+d)^*$ is accepted by $\mathcal{A}$ if and only if it is of the form $a^n c^n$ for some $n \geq 1$. Note that all the other components different than that which starts the computation can reach the final state s_f^3 by reading the empty word.

We analyze now the computation on an input word of the form $a^+b^+(a+b+c+d)^*$. Such a word might lead to acceptance if one component reads completely the prefix of a's while another reads completely the next factor formed by b only. Furthermore, neither a nor d can be the next symbol after the segment of b's. Indeed, an a blocks both these components while a d blocks at least one of them. The analysis may continue in the same way until we conclude that the input word is of the form $a^+b^+c^+d^+$. More precisely, it has to be of the form $a^nb^mc^nd^m$ for some $n, m \geq 1$. Analogously, any input word starting with b that is eventually accepted is either of the form b^md^m or of the form $b^ma^nc^nd^m$ for some $n, m \geq 1$. Consequently,

$$Rec_f(\mathcal{A}) = \{a^nb^mc^nd^m \mid n, m \geq 0\} \cup \{b^ma^nc^nd^m \mid n, m \geq 0\}.$$

Note that every correct input word is actually accepted by means of only two components of $\mathcal{A}$. All the other components reach their final states by just one move when nothing from the input tape is effectively read.

By Lemma 1, since there is no context-free language L and $k \geq 2$ such that $L^k \subseteq Rec_f(\mathcal{A})$, therefore $Rec_f(\mathcal{A})$ cannot lie in $\mathcal{L}_\varepsilon(UDPAS, k)$ for any $k \geq 2$. □

For the rest of this note we shall consider mainly UDPAS accepting by empty stacks. As one can see in Lemma 1, every language accepted by a UPDAS of degree p with empty stacks is a subset of $\text{ш}^p(L)$ for some context-free language L. The following problem naturally arises: When do we have equality? What conditions should L satisfy such that $\text{ш}^p(L)$ is accepted by a UPDAS with empty stacks? It is worth mentioning that for every context-free language L, the language $\text{ш}^p(L)$ is accepted by a UDPAS of degree p with empty stacks if we change the protocol of activating/deactivating the components. More precisely, if we define the language accepted by a UDPAS $\mathcal{A}$ with empty stacks as follows

$$\begin{aligned} Rec_\varepsilon(\mathcal{A}, *) = \{w \mid w \in V^*,\ & (w, q_1, Z_1, q_2, Z_2, \ldots, q_n, Z_n) \vdash_{i_1}^* \\ & (w_1, s_1^{(1)}, \alpha_1^{(1)}, s_2^{(1)}, \alpha_2^{(1)}, \ldots, s_n^{(1)}, \alpha_n^{(1)}) \vdash_{i_2}^* \\ & (w_2, s_1^{(2)}, \alpha_1^{(2)}, s_2^{(2)}, \alpha_2^{(2)}, \ldots, s_n^{(2)}, \alpha_n^{(2)}) \vdash_{i_3}^* \cdots \vdash_{i_m}^* \\ & (\varepsilon, s_1^{(m)}, \varepsilon, s_2^{(m)}, \varepsilon, \ldots, s_n^{(m)}, \varepsilon) \text{ with } m \geq 1, 1 \leq i_1, i_2, \ldots, i_m \leq n \\ & \text{and } s_i^{(m)} \in Q_i, 1 \leq i \leq n\}, \end{aligned}$$

then we can state that $Rec_\varepsilon(\mathcal{A}, *) = \text{ш}^p(L)$, where $\mathcal{A}$ is a UDPAS formed by p copies of the pushdown automaton recognizing L. Therefore, the problem can be reformulated as follows: Can our protocol of activating/deactivating the components lead to more computational power than the protocol just defined above?

We do not have an answer to this problem. However, along the same lines we can show:

Proposition 1. *There are finite languages L such that $\text{ш}^*(L)$ do not belong to $\mathcal{L}_\varepsilon(UDPAS, n)$, for any $n \geq 1$.*

Proof. We take the finite language $\{abc\}$ and prove that

$$\text{ш}^*(abc) = \{w \in \{a,b,c\}^+ \mid |w|_a = |w|_b = |w|_c \ \& \ |x|_a \geq |x|_b \geq |x|_c \text{ for any prefix } x \text{ of } w\},$$

none of the families $\mathcal{L}_\varepsilon(UDPAS, n)$, $n \geq 1$, contains this language.

Assume the contrary, by Lemma 1, there must be a context-free language L such that

$$L^n \subseteq \text{ш}^*(abc) \subseteq \text{ш}^n(L).$$

Let $a^m b^m c^m$, for some $m \geq 1$, be a word in $\text{ш}^*(abc)$; there must exist the words $w_i \in L$, $1 \leq i \leq n$, such that $a^m b^m c^m \in \text{ш}_n(w_1, w_2, \ldots, w_n)$. On the other hand, for any permutation σ of $\{1, 2, \ldots, n\}$ the word $w_{\sigma(1)} w_{\sigma(2)} \ldots w_{\sigma(n)}$ belongs to $\text{ш}^*(abc)$, which means that $w_i \in a^+b^+c^+$ for all i. Furthermore, if $w_i = a^p b^q c^r$, for some $p, q, r \geq 1$, we have $p \geq q \geq r$. We further note that for each $1 \leq i \neq j \leq n \mid \psi(w_i) - \psi(w_j) \mid = (k, k, k)$ holds for some $k \geq 0$. By these considerations and the fact that all words $a^m b^m c^m$, $m \geq 1$, are in $\text{ш}^*(abc)$, we infer that $L \cap a^+b^+c^+$ is an infinite language of the form

$$L \cap a^+b^+c^+ = \{a^{s+k} b^{p+k} c^{q+k} \mid k \in H\},$$

where H is an infinite set of natural numbers. As $L \cap a^+b^+c^+$ is not context-free, it follows that L is not context-free either, which is a contradiction. □

Proposition 2.
1. The family $\mathcal{L}_\varepsilon(UDPAS, n)$, $n \geq 1$, contains semilinear languages only.
2. There are semilinear languages that do not belong to any of these families.

Proof. 1. By Lemma 1, for every UDPAS $\mathcal{A}$ of degree n, $\psi(Rec_\varepsilon(\mathcal{A})) = \psi(L^n)$ holds for some context-free language L, hence $Rec_\varepsilon(\mathcal{A})$ is semilinear.

2. The language considered in Theorem 1 proves the second statement. □

4 Decidability and Closure Properties

Proposition 3. *The emptiness and finiteness problems are decidable for all families $\mathcal{L}_X(UDPAS, n)$, $X \in \{f, \varepsilon\}$, $n \geq 1$.*

Proof. Obviously, the language accepted by a UDPAS is empty/finite if and only if the language accepted by its components is empty/finite. Therefore, the assertion follows from the decidability properties of context-free languages class.

Theorem 2. *None of the families $\mathcal{L}_\varepsilon(UDPAS, n)$ is closed under union and concatenation with singleton languages, union, concatenation, intersection with regular sets, non-erasing morphisms.*

Proof. As we shall see, it suffices to give the reasoning for $n = 2$ only. Let us take the language

$$L = L_1^2 \cup L_2^2 \cup (\text{ш}(L_1, L_2)),$$

where

$$L_1 = \{a^n b^n \mid n \geq 1\}, \qquad L_2 = \{c^m d^m \mid m \geq 1\}.$$

The language L can be accepted by a UDPAS of degree 2. A construction for this system can be easily derived from the definition of the UDPAS in Example 1. We show that $L \cdot \{dcba\}$ cannot be accepted by any UDPAS (no matter its degree) with empty stacks. Assume the contrary, by Lemma 1 there exist a context-free languages E and $k \geq 2$ such that $E^k \subseteq L\{dcba\}$. Therefore, each word in E has to be of the form $xdcba$. We take k words in E, x_1dcba, x_2dcba, ..., x_kdcba such that $x_1dcbax_2dcba \ldots x_kdcba \in L\{dcba\}$, hence $x_1dcbax_2dcba \ldots x_k \in L$ which is a contradiction.

In similar way one can argue that $L \cup \{dcba\}$ does not belong to any family $\mathcal{L}_\varepsilon(UDPAS, n)$.

On the other hand, each regular language $R_k = \{dcba^n \mid n \geq k\}$, $k \geq 2$, belongs to $\mathcal{L}_\varepsilon(UDPAS, k)$. It follows that $\mathcal{L}_\varepsilon(UDPAS, n)$ is not closed under concatenation and union either.

In order to prove the non-closure under intersection with regular sets we return to Example 1. The language accepted by the UPDAS from Example is L_1. But

$$L_1 \cap a^+b^+c^+d^+ = \{a^p b^q c^p d^q \mid p, q \geq 1\},$$

which, by the proof of Theorem 1, does not belong to any family $\mathcal{L}_\varepsilon(DPAS, n)$, $n \geq 2$.

The proof for the non-closure under morphisms is a bit more involved. We consider the language

$$L = \{xxyy, xyxy, xyyx, yyxx, yxyx, yxxy \mid x \in \{a^n\# \mid n \geq 1\}, \\ y \in \{b^n\# \mid n \geq 1\}\}$$

which lies in $\mathcal{L}_\varepsilon(UDPAS, 2)$. The construction of a UDPAS of degree 2 which accepts L by empty stacks is left to the reader. We prove that the language $h(L)$, where $h(a) = h(b) = a$ and $h(\#) = c$, cannot be accepted by any DPAS (not only UDPAS) by empty stacks.

Suppose that $h(L) = Rec_\varepsilon(\mathcal{A})$ with $\mathcal{A} = (\{a, c\}, A_1, A_2, \ldots, A_p)$ for some $p \geq 2$. We may assume that we need at least two components as $h(L)$ is not context-free. There exists a word $z \in Rec_\varepsilon(\mathcal{A})$ such that the following conditions are satisfied with respect to the accepting process of z:

(i) $z = x_1x_2 \ldots x_s$ for some $s > p$, $x_j \in (a + c)^+$.
(ii) For each $1 \leq j \leq s$, the component i_j, $1 \leq i_j \leq p$, is activated when the system starts to read x_j.
(iii) There exist $1 \leq j < t \leq s$ such that $i_j = i_t$ and all numbers $i_{j+1}, \ldots, i_s$ are distinct. That is, for the suffix $x_{j+1} \ldots x_s$ of z each component of $\mathcal{A}$ is activated at most once.

Under these circumstances, the word

$$w = x_1x_2 \ldots x_{j-1}x_jx_tx_{j+1} \ldots x_{t-1}x_{t+1} \ldots x_s$$

is in $Rec_\varepsilon(\mathcal{A})$ as well. If $t \neq s$, then also the word

$$y = x_1 x_2 \dots x_{j-1} x_j x_{j+1} \dots x_{t-1} x_{t+1} \dots x_s x_t$$

is in $Rec_\varepsilon(\mathcal{A})$. Furthermore, the first letter of x_{j+1} is different from the first letter of x_t. There are two possibilities:

(I) $x_{j+1} \in a(a+c)^*, x_t \in c(a+c)^*$.
First, let us note that if x_t ends with a, then $t \neq s$ holds, hence y must be in $h(L)$, But, y ends with a, a contradiction. Therefore, x_t must start and end with c. We note also that t cannot equals s because, if this were the case, then w would ends with a. Then, it follows that y contains two adjacent occurrences of c which is contradictory. Hence, the first case leads to a contradiction.

(II) $x_{j+1} \in c(a+c)^*, x_t \in a(a+c)^*$.
First we note that x_t cannot start and end with a. Indeed, if x_t starts and ends with a, then $t \neq s$ and y ends with a, a contradiction. But if x_t starts with a and ends with c, then w contains two adjacent occurrences of c since the segment $x_j x_t x_{j+1}$ has this property.

Therefore, $h(L) \notin \mathcal{L}_\varepsilon(DPAS, n)$ for any $n \geq 2$, which proves the closure of none of the families $\mathcal{L}_\varepsilon(DPAS, n)$, $n \geq 2$, under morphisms. □

5 Final Remarks

We briefly discuss here a few open problems and possible directions for further developments. We start with the problem formulated in Section 3.

Open Problem 1. *What conditions should a context-free language L satisfy such that $\text{⫫}^p(L)$ is accepted by a UPDAS with empty stacks?*

It is worth mentioning that one can increase the degree of the UPDAS from the proof of Theorem 1 without modifying the accepted language. This is especially due to the fact that the pushdown automaton A recognizes the empty word. Does this hold for any UDPAS accepting the empty word? Which is the situation for acceptance with empty stacks? More generally,

Open Problem 2. *Is there any hierarchy depending on the number of components?*

However, if the classes of languages accepted by UPDAS with empty stacks form a hierarchy depending on the number of components, then this hierarchy is necessarily infinite.

Theorem 3. *There exist arbitrarily many natural numbers n such that*

$$\mathcal{L}_\varepsilon(UPDAS, n) \setminus \mathcal{L}_\varepsilon(UPDAS, k) \neq \emptyset.$$

Proof. Let A be the pushdown automaton accepting the language

$$L = \{\#a^m b^m \$ \mid m \geq 1\} \cup \{\#c^m d^m \$ \mid m \geq 1\},$$

and $\mathcal{A}$ be the UPDAS formed by n copies of A, where n is a prime number. We first note that $Rec_\varepsilon(\mathcal{A})$ is not a context-free language. Indeed,

$$Rec_\varepsilon(\mathcal{A}) \cap \#^n a^+ c^+ b^+ d^+ \$^n = \{\#^n a^p c^q b^p d^q \$^n \mid p, q \geq 1, p + q \geq n\},$$

hence $Rec_\varepsilon(\mathcal{A})$ cannot be context-free. We now claim that $Rec_\varepsilon(\mathcal{A})$ cannot be accepted by any UPDAS of a degree inferior to n. Assume the contrary and let $\mathcal{A}'$ be a UPDAS of degree $k < n$ such that $Rec_\varepsilon(\mathcal{A}) = Rec_\varepsilon(\mathcal{A}')$. By Lemma 1, there exists a context-free language R such that

$$R^k \subseteq Rec_\varepsilon(\mathcal{A}) = Rec_\varepsilon(\mathcal{A}') \subseteq \text{ш}^k(R).$$

Clearly, for every word $w \in Rec_\varepsilon(\mathcal{A})$, $|w|_\# = |w|_\$ = n$ holds. Therefore, for any word $x \in R$, $|x|_\# = |x|_\$ = p$, with $kp = n$ must hold. This implies that $k = 1$, hence $R = Rec_\varepsilon(\mathcal{A})$ which is a contradiction. □

As we have seen the emptiness and finiteness problems are decidable for UDPAS accepting by empty stacks and the complexity of these problems is directly derived from the complexity of the same problems for usual pushdown automata. The situation seems to be different for the membership problem. We recall that for the shuffling of two context-free languages, the non-uniform version of the membership problem is already NP-hard [1]. However, we ask:

Open Problem 3. *Which is the complexity of the membership problem for UPDAS accepting with empty stacks?*

We have proved that UDPAS accepting by final states are strictly more powerful than UDPAS accepting by empty stacks. In our view, the classes $\mathcal{L}_f(DPAS, n)$, $n \geq 1$, deserve to be further investigated.

Last but not least, the deterministic variants of the automata systems considered here appear to be attractive.

References

1. Berglund, M., Björklund, H., Högberg, J.: Recognizing Shuffled Languages. In: Dediu, A.-H., Inenaga, S., Martín-Vide, C. (eds.) LATA 2011. LNCS, vol. 6638, pp. 142–154. Springer, Heidelberg (2011)
2. Bordihn, H., Kutrib, M., Malcher, A.: Undecidability and hierarchy results for parallel communicating finite automata. Int. J. Found. Comput. Sci. 22, 1577–1592 (2011)
3. Choudhary, A., Krithivasan, K., Mitrana, V.: Returning and non-returning parallel communicating finite automata are equivalent. Information Processing Letters 41, 137–145 (2007)
4. Csuhaj-Varju, E., Dassow, J.: On cooperating distributed grammar systems. J. Inform. Process. Cybern., EIK 26, 49–63 (1990)
5. Csuhaj-Varju, E., Dassow, J., Kelemen, J., Păun, G.: Grammar Systems. A grammatical approach to distribution and cooperation. Gordon and Breach (1994)
6. Csuhaj-Varju, E., Martín-Vide, C., Mitrana, V., Vaszil, G.: Parallel communicating pushdown automata systems. Int. J. Found. Comput. Sci. 11, 633–650 (2000)

7. Csuhaj-Varju, E., Mitrana, V., Vaszil, G.: Distributed Pushdown Automata Systems: Computational Power. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 218–229. Springer, Heidelberg (2003)
8. Dassow, J., Păun, G., Rozenberg, G.: Grammar systems. In: [16], vol. 2
9. Dassow, J., Mitrana, V.: Stack cooperation in multi-stack pushdown automata. J. Comput. System Sci. 58, 611–621 (1999)
10. Durfee, E.H., et al.: Cooperative distributed problem solving. In: Barr, A., Cohen, P.R., Feigenbaum, E.A. (eds.) The Handbook of AI, vol. 4. Addison-Wesley, Reading (1989)
11. Krithivasan, K., Sakthi Balan, M., Harsha, P.: Distributed processing in automata. Int. J. Found. Comput. Sci. 10, 443–464 (1999)
12. Martín-Vide, C., Mitrana, V.: Some undecidable problems for parallel communicating finite automata systems. Information Processing Letters 77, 239–245 (2001)
13. Martín-Vide, C., Mateescu, A., Mitrana, V.: Parallel finite automata systems communicating by states. Int. J. Found. Comput. Sci. 13, 733–749 (2002)
14. Meersman, R., Rozenberg, G.: Cooperating Grammar Systems. In: Winkowski, J. (ed.) MFCS 1978. LNCS, vol. 64, pp. 364–374. Springer, Heidelberg (1978)
15. Păun, G., Sântean, L.: Parallel communicating grammar systems: the regular case. Ann. Univ. Bucharest, Ser. Matem.-Inform. 38, 55–63 (1989)
16. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages, vol. 1-3. Springer, Berlin (1997)