

# A Note on Limited Pushdown Alphabets in Stateless Deterministic Pushdown Automata

Tomáš Masopust

Institute of Mathematics, Academy of Sciences of the Czech Republic  
Žitkova 22, 616 62 Brno, Czech Republic  
masopust@math.cas.cz

**Abstract.** Recently, an infinite hierarchy of languages accepted by stateless deterministic pushdown automata has been established based on the number of pushdown symbols. However, the witness language for the  $n$ th level of the hierarchy is over an input alphabet with  $2(n-1)$  elements. In this paper, we improve this result by showing that a binary alphabet is sufficient to establish this hierarchy. As a consequence of our construction, we solve the open problem formulated by Meduna et al. Then we extend these results to  $m$ -state realtime deterministic pushdown automata, for all  $m \geq 1$ . The existence of such a hierarchy for  $m$ -state deterministic pushdown automata is left open.

## 1 Introduction

Pushdown automata, and especially deterministic pushdown automata, play a central role in many applications of context-free languages. Considering state complexity, it is well-known that any pushdown automaton can be transformed to an equivalent pushdown automaton with a single state. Thus, the hierarchy based on the number of states collapses to only one level. In that case, the sole state is not important and one-state automata are also called stateless in the literature. On the other hand, however, no such transformation is possible for deterministic pushdown automata since it is shown in [7] that the number of states in deterministic pushdown automata establishes an infinite state hierarchy on the family of deterministic context-free languages. The reader is referred to [7] for more details, including binary witness languages.

Recently, Meduna et al. [17] have established an infinite hierarchy of languages on the lowest level of the state hierarchy, on the level of stateless deterministic pushdown automata, based on the number of pushdown symbols. They have shown that there exist languages accepted by stateless deterministic pushdown automata with  $n$  pushdown symbols that cannot be accepted by any stateless deterministic pushdown automaton with  $n-1$  pushdown symbols. However, the size of the input alphabet of the witness language for each level of the hierarchy is almost twice bigger than the size of the pushdown alphabet, which is linear with respect to the level of the hierarchy. More specifically, they constructed an infinite sequence of languages  $L_n$ , for all  $n \geq 1$ , such that the alphabet of the language  $L_n$  is of cardinality  $2n$ , and they proved that any stateless deterministic pushdown automaton accepting the language  $L_n$  requires at least  $n+1$  pushdown symbols. Thus, their construction requires a growing alphabet and it was left

open whether the hierarchy can be shown with witness languages over a fixed alphabet. In addition, they formulated an open question of what is the role of non-input pushdown symbols in this hierarchy, that is, can a similar infinite hierarchy be established based on the number of non-input pushdown symbols?

In this paper, we improve their result by constructing a sequence of witness languages over a binary alphabet, which is the best possible improvement because any unary language accepted by a stateless deterministic pushdown automaton is a singleton. In other words, languages accepted by deterministic pushdown automata (by empty pushdown) are prefix-free [7]. As an immediate consequence of our construction, we obtain a solution to the open problem. Then we show that a similar idea can be used for realtime pushdown automata to establish infinite hierarchies based on the number of pushdown symbols on each level of the state hierarchy. Again, the witness languages are binary. However, the case of general deterministic pushdown automata is left open.

Stateless automata of many types are of great interest and have widely been investigated in the literature. For instance, stateless multicounter machines have been studied in [2], hierarchies of stateless multicounter  $5' \rightarrow 3'$  Watson-Crick automata have been investigated in [1,18], multihead finite and pushdown automata have been studied in [4,9], stateless restarting automata in [11,12], and stateless automata and their relationship to P systems in [24]. Pushdown store languages, that is, languages consisting of strings occurring on the pushdown along accepting computations of a pushdown automaton, have been studied in [15,16], including the languages of stateless pushdown automata. Some decision results concerning stateless and deterministic pushdown automata can be found in [5,22].

## 2 Preliminaries

In this paper, we assume that the reader is familiar with automata and formal language theory [7,8,19]. For a set  $A$ ,  $|A|$  denotes the cardinality of  $A$ , and  $2^A$  denotes the powerset of  $A$ . An alphabet is a finite nonempty set. For an alphabet  $V$ ,  $V^*$  represents the free monoid generated by  $V$ , where the unit (the empty string) is denoted by  $\varepsilon$ .

A *pushdown automaton* (PDA) is a septuple  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , where  $Q$  is the finite set of states,  $\Sigma$  is the input alphabet,  $\Gamma$  is the pushdown alphabet,  $\delta$  is the transition function from  $Q \times \Gamma \times (\Sigma \cup \{\varepsilon\})$  to the set of finite subsets of  $Q \times \Gamma^*$ ,  $q_0 \in Q$  is the initial state,  $Z_0 \in \Gamma$  is the initial pushdown symbol, and  $F \subseteq Q$  is the set of accepting states. A configuration of  $M$  is any element of  $Q \times \Gamma^* \times \Sigma^*$ . For a configuration  $(q, \gamma, \sigma)$ ,  $q$  denotes the current state of  $M$ ,  $\gamma$  denotes the content of the pushdown (with the top as the rightmost symbol), and  $\sigma$  denotes the unread part of the input string. The automaton changes the configuration according to the transition function  $\delta$ . This one-step relation is denoted by  $\vdash$  and defined so that  $(q, \gamma A, a\sigma) \vdash (p, \gamma w, \sigma)$  if  $(p, w)$  is in  $\delta(q, A, a)$ , where  $\gamma$  and  $w$  are strings of pushdown symbols,  $A$  is a pushdown symbol,  $a$  is an input symbol or  $\varepsilon$ , and  $\sigma$  is a string of input symbols. Let  $\vdash^*$  denote the reflexive and transitive closure of the relation  $\vdash$ . The language accepted by  $M$  (by a final state and empty pushdown) is denoted by  $L(M)$  and defined as  $L(M) = \{w \in \Sigma^* \mid (q_0, Z_0, w) \vdash^* (f, \varepsilon, \varepsilon) \text{ for some } f \in F\}$ .

Automaton  $M$  is *deterministic* (DPDA) if there is no more than one move the automaton can make from any configuration, that is,  $|\delta(q, A, a)| \leq 1$ , for any state  $q$ , pushdown symbol  $A$ , and input symbol  $a$ , and if  $\delta(q, A, \varepsilon)$  is defined, that is,  $\delta(q, A, \varepsilon) \neq \emptyset$ , then  $\delta(q, A, a)$  is not defined for any input symbol  $a$ .

Automaton  $M$  is *realtime* (RPDA) if it is deterministic and if no  $\varepsilon$ -transitions are defined, that is, if  $\delta(q, A, a)$  is defined, then  $a \neq \varepsilon$ .

Automaton  $M$  is *stateless* (SPDA) if it has only one state, that is,  $|Q| = 1$ . Moreover, in stateless pushdown automata, we allow an initial pushdown string  $\alpha \in \Gamma^+$  instead of an initial pushdown symbol. Thus, in this case, we simply write  $M = (\Sigma, \Gamma, \delta, \alpha)$ . Note that unlike the general case of deterministic pushdown automata, in the case of stateless deterministic pushdown automata, the realtime restriction does not decrease the power of the machine [7].

In what follows, the notation  $(q, \gamma) \xrightarrow{a} (p, \gamma')$ , for  $a$  in  $\Sigma$  and  $\gamma, \gamma'$  in  $\Gamma^*$ , denotes the computational step  $(q, \gamma, a) \vdash (p, \gamma', \varepsilon)$ , and for a string  $w \in \Sigma^*$ , the notation  $(q, \gamma) \xrightarrow{w}^* (p, \gamma')$  denotes the maximal computation  $(q, \gamma, w) \vdash^* (p, \gamma', \varepsilon)$ , that is, no other  $\varepsilon$ -transitions are possible from the configuration  $(p, \gamma', \varepsilon)$ . For stateless automata, the state is eliminated from this notation.

We now recall the definition of families of languages accepted by stateless deterministic pushdown automata with  $n$  pushdown symbols.

**Definition 1.** A stateless deterministic pushdown automaton  $M = (\Sigma, \Gamma, \delta, \alpha)$  is *n-pushdown-alphabet-limited*, for  $n \geq 1$ , if  $|\Gamma| \leq n$ . The family *n-SDPDA* denotes all languages accepted by an *n-pushdown-alphabet-limited stateless deterministic pushdown automaton*. Similarly, the family *n-SRPDA* denotes all languages accepted by an *n-pushdown-alphabet-limited stateless realtime pushdown automaton*.

### 3 Main results

Recall that it is known [7] that any language  $L$  accepted by a stateless deterministic pushdown automaton is prefix-free. Thus, if  $L$  contains  $\varepsilon$ , then  $L = \{\varepsilon\}$ . It is also known that any stateless deterministic pushdown automaton accepting a language different from  $\{\varepsilon\}$  can be transformed to an equivalent stateless realtime pushdown automaton. To simplify proves, we extend this result by showing that the resulting stateless realtime pushdown automaton does not require any new pushdown symbol.

**Theorem 1.** For any stateless deterministic pushdown automaton with  $n$  pushdown symbols accepting a language  $L \neq \{\varepsilon\}$ , there exists an equivalent stateless realtime pushdown automaton with no more than  $n$  pushdown symbols.

*Proof.* Assume that all pushdown symbols appear in an accepting computation. Let  $X \xrightarrow{\varepsilon} \sigma$  be an  $\varepsilon$ -transition of  $M = (\Sigma, \Gamma, \delta, \alpha)$ . If  $X$  appears in  $\sigma$ , then any computation reaching  $X$  on top of the pushdown is not accepting because  $X$  cannot be eliminated from the pushdown; it is always replaced with  $\sigma$  containing  $X$  again, which contradicts our assumption. Modify the automaton  $M$  by replacing all occurrences of  $X$  on right-hand sides of transitions with  $\sigma$ , and by removing  $X \xrightarrow{\varepsilon} \sigma$  from  $\delta$  and  $X$  from  $\Gamma$ . The accepted language is not changed because the only action of  $X$  was to be replaced with

$\sigma$ , and the resulting automaton has one  $\varepsilon$ -transition less. Repeating this substitution results in an equivalent stateless realtime pushdown automaton with no more than  $n$  pushdown symbols.

This result can be rewritten as follows.

**Corollary 1.** *For all  $n \geq 1$ ,  $n$ -SDPDA =  $n$ -SRPDA.*

In the following subsection, we improve the result presented in [17] by giving binary witness languages and answering the open problem formulated there.

### 3.1 Binary alphabet

First, note that from the fact that any language accepted by a stateless deterministic pushdown automaton is prefix-free, any such a unary language is of the form  $a^n$ , for some  $n \geq 0$ . Every such language can be accepted by a stateless deterministic pushdown automaton with only one pushdown symbol (equal to the input symbol) starting from the initial string  $a^n$  with  $\delta(a, a) = \varepsilon$ . Thus, to establish the infinite hierarchy based on the number of pushdown symbols using only a binary alphabet is the best possible improvement that can be done. To do this, we define the following sequence of languages

$$\begin{aligned} L_1 &= \{a^c\}, \text{ for some } c \geq 0, \text{ and} \\ L_n &= \{b^k a \mid 1 \leq k \leq n-1\}, \text{ for any } n \geq 2. \end{aligned} \tag{1}$$

We now prove that for all  $n \geq 1$ , any stateless deterministic pushdown automaton accepting the language  $L_n$  requires at least  $n$  pushdown symbols. First, we show that  $n$  pushdown symbols is sufficient.

**Lemma 1.** *For any  $n \geq 1$ , there exists a stateless deterministic pushdown automaton with  $n$  pushdown symbols accepting the language  $L_n$  defined in (1).*

*Proof.* As mentioned above, there exists a stateless deterministic pushdown automaton with one pushdown symbol accepting the language  $L_1$ . Thus, consider the language  $L_n$  for  $n \geq 2$ . This language is accepted by the stateless pushdown automaton  $M_n = (\{a, b\}, \{X_0, X_1, \dots, X_{n-1}\}, \delta, X_0)$ , where the transition function  $\delta$  is defined as follows:

1.  $\delta(X_i, b) = X_{i+1}$ , for  $0 \leq i \leq n-2$ ,
2.  $\delta(X_i, a) = \varepsilon$ , for  $1 \leq i \leq n-1$ .

Symbols  $X_1, X_2, \dots, X_n$  are used to count the number of  $b$ 's and to restrict this number to  $n-1$ . Notice that  $M_n$  is deterministic (realtime) and that  $|\Gamma| = n$ .

We next prove that at least  $n$  pushdown symbols are necessary for the language  $L_n$ .

**Lemma 2.** *For any  $n \geq 1$ , there does not exist a stateless deterministic pushdown automaton accepting the language  $L_n$  defined in (1) with  $n-1$  pushdown symbols.*

*Proof.* The case  $n = 1$  is trivial and follows from the definition. Thus, consider the case  $n \geq 2$ . Let  $M_n = (\{a, b\}, \Gamma, \delta, \alpha)$  be a stateless deterministic pushdown automaton accepting the language  $L_n$ . By Theorem 1, we can assume that  $M_n$  is realtime. Consider the computations on strings  $b^k a$  for all  $k = 1, 2, \dots, n - 1$ . Then

$$\alpha \xrightarrow{b} \pi_1 X_1 \xrightarrow{b} \pi_2 X_2 \xrightarrow{b} \dots \xrightarrow{b} \pi_{n-1} X_{n-1} \xrightarrow{a}^* \varepsilon.$$

As symbol  $a$  is accepted from all configurations  $\pi_k X_k$ , it implies that  $X_k \xrightarrow{a} \varepsilon$  and  $\pi_k = \varepsilon$ , for all  $k = 1, 2, \dots, n - 1$ , that is,

$$\alpha \xrightarrow{b} X_1 \xrightarrow{b} X_2 \xrightarrow{b} \dots \xrightarrow{b} X_{n-1} \xrightarrow{a} \varepsilon.$$

Assume that there exist  $i < j$  such that  $X_i = X_j$ . Since  $X_i \xrightarrow{b^{n-1-i} a}^* \varepsilon$ , the automaton accepts the string  $b^{j+n-1-i} a$  that does not belong to the language  $L_n$  because  $n - 1 + (j - i) > n - 1$ . Hence, all the pushdown symbols  $X_i$ , for  $i = 1, 2, \dots, n - 1$ , are pairwise different. It remains to show that the automaton needs at least one more pushdown symbol. To do this, consider the form of the initial string  $\alpha$ . If  $\alpha \in \{X_1, X_2, \dots, X_{n-1}\}^+$ , then  $\alpha \xrightarrow{a^{|\alpha|}}^* \varepsilon$  because  $X_i \xrightarrow{a} \varepsilon$ , for all  $i = 1, 2, \dots, n - 1$ . Thus, the automaton needs at least  $n$  pushdown symbols.

As a consequence of the previous two lemmas, we have the following result. The strictness of the inclusion has been shown in [17]. We have added binary witness languages.

**Theorem 2.** *For each  $n \geq 1$ ,  $n$ -SDPDA  $\subsetneq (n + 1)$ -SDPDA. In addition, the witness languages are binary.*

### 3.2 Non-input pushdown symbols

In [17], the authors formulated a question of whether an infinite hierarchy can be established based on the number of non-input pushdown symbols, that is, whether there exists a similar hierarchy when Definition 1 is modified as follows.

**Definition 2.** *A stateless deterministic pushdown automaton  $M = (\Sigma, \Gamma, \delta, \alpha)$  is an  $n$ -non-input-pushdown-alphabet-limited, for  $n \geq 1$ , if the number of non-input pushdown symbols is limited by  $n$ , that is, if  $|\Gamma| - |\Sigma| \leq n$ . The family  $n$ -niSDPDA denotes all languages accepted by an  $n$ -non-input-pushdown-alphabet-limited stateless deterministic pushdown automaton.*

As an immediate consequence of the previous results, we get the answer to this question.

**Theorem 3.** *For each  $n \geq 0$ ,  $n$ -niSDPDA  $\subsetneq (n + 1)$ -niSDPDA.*

*Proof.* We define languages  $K_n = L_{n+2}$ . Then any stateless deterministic pushdown automaton accepting the language  $K_n$  requires  $n + 2$  pushdown symbols. Since the input alphabet is binary, we immediately get  $|\Gamma| - |\Sigma| = n$ , as required.

## 4 Generalization to realtime pushdown automata

It is a natural question to ask whether a similar hierarchy can be obtained for the other levels of the state hierarchy. To prove this for realtime pushdown automata, we define the following sequence of languages

$$\begin{aligned} L_{m,1} &= \{a^c\}, \text{ for some } c \geq 0, \text{ and} \\ L_{m,n} &= \{b^k a \mid 1 \leq k \leq mn - 1\}, \text{ for any } m \geq 1 \text{ and } n \geq 2. \end{aligned} \quad (2)$$

We now prove that for all  $m, n \geq 1$ , any  $m$ -state deterministic pushdown automaton accepting the language  $L_{m,n}$  requires at least  $n$  pushdown symbols. The sufficiency is shown first.

**Lemma 3.** *For any  $m, n \geq 1$ , there exists an  $m$ -state realtime pushdown automaton with  $n$  pushdown symbols accepting the language  $L_{m,n}$  defined in (2).*

*Proof.* Consider the language  $L_{m,n}$  for  $m \geq 1$  and  $n \geq 2$ . This language is accepted by the  $m$ -state deterministic pushdown automaton

$$M_{m,n} = (\{q_0, q_1, \dots, q_{m-1}\}, \{a, b\}, \{X_0, X_1, \dots, X_{n-1}\}, \delta, q_0, X_0, \{q_{m-1}\}),$$

where the transition function  $\delta$  is defined as follows:

1.  $\delta(q_j, X_i, b) = (q_j, X_{i+1})$ , for  $0 \leq j \leq m-1$  and  $0 \leq i \leq n-2$ ,
2.  $\delta(q_j, X_{n-1}, b) = (q_{j+1}, X_0)$ , for  $0 \leq j \leq m-2$ ,
3.  $\delta(q_0, X_i, a) = (q_{m-1}, \varepsilon)$ , for  $1 \leq i \leq n-1$ ,
4.  $\delta(q_j, X_i, a) = (q_{m-1}, \varepsilon)$ , for  $1 \leq j \leq m-1$  and  $0 \leq i \leq n-1$ .

Pushdown symbols in combination with states are used to count the number of  $b$ 's and to restrict this number to  $mn - 1$ . Notice that  $M_{m,n}$  is realtime and that  $|\Gamma| = n$ .

We now prove that at least  $n$  pushdown symbols are necessary for any  $m$ -state realtime pushdown automaton to accept the language  $L_{m,n}$ .

**Lemma 4.** *For any  $m, n \geq 1$ , there does not exist any  $m$ -state realtime pushdown automaton accepting the language  $L_{m,n}$  defined in (2) with  $n - 1$  pushdown symbols.*

*Proof.* Let  $m \geq 1$  be fixed, but arbitrary. The case  $n = 1$  is trivial and follows from the definition. Thus, consider the case  $n \geq 2$ . For the sake of contradiction, assume that there exists an  $m$ -state realtime pushdown automaton  $M_{m,n} = (\{q_0, q_1, \dots, q_{m-1}\}, \{a, b\}, \Gamma, \delta, q_0, X_0, F)$  with  $n - 1$  pushdown symbols accepting the language  $L_{m,n}$ . Consider the computations on strings  $b^k a$  for all  $k = 1, 2, \dots, mn - 1$ . Then

$$\begin{aligned} (q_0, X_0) &\xrightarrow{b} (q_1, \pi_1 X_1) \xrightarrow{b} (q_2, \pi_2 X_2) \xrightarrow{b} \dots \\ &\dots \xrightarrow{b} (q_{mn-1}, \pi_{mn-1} X_{mn-1}) \xrightarrow{a} (q_f, \varepsilon), \end{aligned}$$

where  $q_f \in F$ . As symbol  $a$  is accepted from all configurations  $(q_k, \pi_k X_k)$ , it implies that  $(q_k, X_k) \xrightarrow{a} (q_{f_k}, \varepsilon)$ , for some  $q_{f_k} \in F$ , and  $\pi_k = \varepsilon$ , for all  $k = 1, 2, \dots, mn - 1$ , that is,

$$(q_0, X_0) \xrightarrow{b} (q_1, X_1) \xrightarrow{b} (q_2, X_2) \xrightarrow{b} \dots \xrightarrow{b} (q_{mn-1}, X_{mn-1}) \xrightarrow{a} (q_f, \varepsilon).$$

As we have  $m$  states and  $n - 1$  pushdown symbols, there must exist  $i < j$  such that  $(q_i, X_i) = (q_j, X_j)$ . Since  $(q_i, X_i) \xrightarrow{b^{mn-1-i}a}^* \varepsilon$ , the automaton accepts string  $b^{j+mn-1-i}a$  that does not belong to the language  $L_{m,n}$  because  $mn - 1 + (j - i) > mn - 1$ , which is a contradiction. Thus, the automaton needs at least  $n$  pushdown symbols.

We now present definitions of families of languages accepted by  $m$ -state deterministic pushdown automata with  $n$  pushdown symbols.

**Definition 3.** A pushdown automaton  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  is  $n$ -pushdown-alphabet-limited, for  $n \geq 1$ , if  $|\Gamma| \leq n$ . The family  $(m, n)$ -RPDA denotes all languages accepted by an  $m$ -state realtime pushdown automaton with  $n$  pushdown symbols.

**Definition 4.** A pushdown automaton  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  is an  $n$ -non-input-pushdown-alphabet-limited, for  $n \geq 1$ , if  $|\Gamma| - |\Sigma| \leq n$ . The family  $(m, n)$ -niRPDA denotes all languages accepted by an  $m$ -state realtime pushdown automaton with  $n$  non-input pushdown symbols.

As a consequence of the previous two lemmas, we have the following results.

**Theorem 4.** For each  $m, n \geq 1$ ,  $(m, n)$ -RPDA  $\subsetneq (m, n + 1)$ -RPDA. In addition, the witness languages are binary.

**Theorem 5.** For each  $m \geq 1$  and  $n \geq 0$ ,  $(m, n)$ -niRPDA  $\subsetneq (m, n + 1)$ -niRPDA.

*Proof.* We define languages  $K_{m,n} = L_{m,n+2}$ . Then any  $m$ -state realtime pushdown automaton accepting the language  $K_{m,n}$  requires  $n + 2$  pushdown symbols. Since the input alphabet is binary, we immediately get  $|\Gamma| - |\Sigma| = n$ , as required.

Finally, note that Theorem 1 does not hold for deterministic pushdown automata that are not stateless. The following example shows that there exists a two-state deterministic automaton with two pushdown symbols accepting the language  $L_{m,n}$ , for all  $m, n \geq 1$ . This implies that to establish a similar infinite hierarchy for deterministic pushdown automata based on the number of states and pushdown symbols over a binary (constant) input alphabet requires more sophisticated approaches.

*Example 1.* Let  $m, n \geq 1$  be fixed, but arbitrary. Then the automaton  $M = (\{f, q\}, \{a, b\}, \{Z, B\}, \delta, f, Z, \{f\})$ , where  $\delta$  is defined so that

1.  $\delta(f, Z, b) = (q, B^{mn-1})$ ,
2.  $\delta(q, B, b) = (q, \varepsilon)$ ,
3.  $\delta(q, B, a) = (f, \varepsilon)$ ,
4.  $\delta(f, B, \varepsilon) = (f, \varepsilon)$ ,

accepts the language  $L_{m,n}$ . The automaton reads the first input symbol  $b$ , changes its state, and pushes a string of  $mn - 1$  symbols  $B$  to the pushdown. Then it compares input symbol  $b$  against  $B$  on the pushdown until it reads symbol  $a$  with  $B$  on the pushdown. In that case, it goes to the other state where it empties the whole pushdown without reading any other input symbol.

## 5 Conclusion

It is a natural question to ask whether a similar hierarchy can be obtained for deterministic pushdown automata. It is likely that such a hierarchy exists on each level, however, we have not established it in this paper. Can such a hierarchy be established with witness languages over a binary (constant) alphabet?

Languages accepted by stateless deterministic pushdown automata are also called *simple languages* because they are generated by so-called *simple grammars* [7]. It was shown in [10] that the equivalence problem for simple grammars is decidable, which is, together with the general result by Sénizergues [20,21], in contrast to the fact that containment is undecidable even for simple languages as shown in [3]. It is remarkable that the best known algorithm for the equivalence of simple languages works in time  $O(n^6 \text{ polylog } n)$ , where  $n$  is the size of the grammar, see [13]. Hence, simple languages are in some sense the simplest languages for which containment is undecidable. Note that further simplification results in so-called *very simple languages*, for which the containment was shown decidable in [23], and the algorithm improved and simplified in [14]. Simple languages also play a role in process algebras, see, e.g., [6].

Finally, note that it is unsolvable to decide whether for a context-free language  $L$  there exists an integer  $n$  such that  $L$  is accepted by a stateless deterministic pushdown automaton with a pushdown alphabet of cardinality  $n$ . This follows immediately from the undecidability of the problem whether a context-free language  $L$  can be accepted by a stateless deterministic pushdown automaton, see [7] where this problem is formulated as an exercise. As far as the author knows, it is also an open problem whether there exists an algorithm to compute, for a given deterministic context-free language  $L$ , the minimal  $n$  such that  $L$  belongs to the  $n$ th level of the state hierarchy.

## Acknowledgement.

Research supported by the GAČR grant no. P202/11/P028 and by RVO: 67985840.

## References

1. O. Egecioğlu, L. Hegedüs, and B. Nagy. Hierarchies of stateless multicounter  $5' \rightarrow 3'$  Watson-Crick automata languages. *Fund. Inform.*, 110(1-4):111–123, 2011.
2. O. Egecioğlu and O. H. Ibarra. On stateless multicounter machines. In K. Ambos-Spies, B. Löwe, and W. Merkle, editors, *Proc. of CiE 2009*, volume 5635 of *LNCS*, pages 178–187. Springer, Heidelberg, 2009.
3. E. P. Friedman. The inclusion problem for simple languages. *Theoret. Comput. Sci.*, 1:297–316, 1976.
4. P. Frisco and O. H. Ibarra. On stateless multihead finite automata and multihead pushdown automata. In V. Diekert and D. Nowotka, editors, *Proc. of DLT 2009*, volume 5583 of *LNCS*, pages 240–251. Springer, Heidelberg, 2009.
5. J. H. Gallier. DPDA's in 'atomic normal form' and applications to equivalence problems. *Theoret. Comput. Sci.*, 14(2):155–186, 1981.
6. J. F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Inform. and Comput.*, 115(2):354–371, 1994.



7. M. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Boston, USA, 1978.
8. J. E. Hopcroft and J. D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley, Boston, USA, 1969.
9. O. H. Ibarra, J. Karhumäki, and A. Okhotin. On stateless multihead automata: Hierarchies and the emptiness problem. In E. S. Laber, C. Bornstein, L. T. Nogueira, and L. Faria, editors, *Proc. of LATIN 2008*, volume 4957 of *LNCS*, pages 94–105. Springer, Heidelberg, 2008.
10. A. Korenjak and J. Hopcroft. Simple deterministic languages. In *Proc. of SWAT (FOCS)*, pages 36–46, Berkeley, California, USA, 1966.
11. M. Kutrib, H. Messerschmidt, and F. Otto. On stateless deterministic restarting automata. *Acta Inform.*, 47(7):391–412, 2010.
12. M. Kutrib, H. Messerschmidt, and F. Otto. On stateless two-pushdown automata and restarting automata. *Internat. J. Found. Comput. Sci.*, 21(5):781–798, 2010.
13. S. Lasota and W. Rytter. Faster algorithm for bisimulation equivalence of normed context-free processes. In R. Kráľovič and P. Urzyczyn, editors, *Proc. of MFCS 2006*, volume 4162 of *LNCS*, pages 646–657. Springer, Heidelberg, 2006.
14. E. Mäkinen. On the inclusion problem for very simple deterministic pushdown automata. *J. Autom. Lang. Comb.*, 6(3):351–357, 2001.
15. A. Malcher, K. Meckel, C. Mereghetti, and B. Palano. Descriptive complexity of pushdown store languages. In M. Kutrib, N. Moreira, and R. Reis, editors, *Proc. of DCFCS 2012*, volume 7386 of *LNCS*, pages 209–221. Springer, Heidelberg, 2012.
16. A. Malcher, K. Meckel, C. Mereghetti, and B. Palano. Descriptive complexity of pushdown store languages. IFIG Research Report 1203, Institut für Informatik, Universität Giessen, Germany, 2012.
17. A. Meduna, L. Vrabel, and P. Zemek. An infinite hierarchy of language families resulting from stateless pushdown automata with limited pushdown alphabets. In M. Kutrib, N. Moreira, and R. Reis, editors, *Proc. of DCFCS 2012*, volume 7386 of *LNCS*, pages 236–243. Springer, Heidelberg, 2012.
18. B. Nagy, L. Hegedüs, and O. Eğecioğlu. Hierarchy results on stateless multcounter  $5' \rightarrow 3'$  Watson-Crick automata. In J. Cabestany, I. Rojas, and G. Joya, editors, *Proc. of IWANN 2011, Part I*, volume 6691 of *LNCS*, pages 465–472. Springer, Heidelberg, 2011.
19. A. Salomaa. *Formal languages*. Academic Press, New York, 1973.
20. G. Sénizergues.  $T(A)=T(B)$ ? In *Proc. of ICALP 1999*, volume 1644 of *LNCS*. Springer, Heidelberg, 1999.
21. G. Sénizergues.  $L(A)=L(B)$ ? decidability results from complete formal systems. *Theoret. Comput. Sci.*, 251(1-2):1–166, 2001.
22. L. G. Valiant. Decision procedures for families of deterministic pushdown automata. Technical Report CS-RR-001, Department of Computer Science, University of Warwick, Coventry, UK, 1973.
23. M. Wakatsuki and E. Tomita. A fast algorithm for checking the inclusion for very simple deterministic pushdown automata. *IEICE Trans. Inform. Sys.*, E76-D:1224–1233, 1993.
24. L. Yang, Z. Dang, and O. H. Ibarra. On stateless automata and P systems. *Internat. J. Found. Comput. Sci.*, pages 1259–1276, 2008.