I F I G
Research
Report

Institut für Informatik

Descriptional Complexity of Pushdown Store Languages

Andreas Malcher Katja Meckel
Carlo Mereghetti Beatrice Palano

IFIG Research Report 1203
May 2012

Institut für Informatik
JLU Gießen
Arndtstraße 2
35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

IFIG Research Report

IFIG Research Report 1203, May 2012

Descriptional Complexity of Pushdown Store Languages[1]

Andreas Malcher[2] and Katja Meckel[3]

Institut für Informatik, Universität Giessen
Arndtstraße 2, 35392 Giessen, Germany

and

Carlo Mereghetti[4] and Beatrice Palano[5]

Dipartimento di Informatica, Università degli Studi di Milano
via Comelico 39/41, 20135 Milano, Italy

Abstract. It is well known that the *pushdown store language* $P(M)$ of a pushdown automaton (PDA) M — i.e., the language consisting of words occurring on the pushdown along accepting computations of M — is a regular language. Here, we design *succinct nondeterministic finite automata (NFA)* accepting $P(M)$. In detail, an upper bound on the size of an NFA for $P(M)$ is obtained, which is quadratic in the number of states and linear in the number of pushdown symbols of M. Moreover, this upper bound is shown to be asymptotically optimal. Then, several restricted variants of PDA are considered, leading to improved constructions. In all cases, we prove the asymptotical optimality of the size of the resulting NFA. Finally, we apply our results to decidability questions related to PDA, and obtain solutions in deterministic polynomial time.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Relations between models*; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Relations among complexity classes*; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs between Complexity Measures; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*;

Additional Key Words and Phrases: pushdown automata; pushdown store languages; descriptional complexity; decidability questions.

---

[1]Partially supported by CRUI/DAAD under the project "Programma Vigoni: Descriptional Complexity of Non-Classical Computational Models."

[2]E-mail: malcher@informatik.uni-giessen.de
[3]E-mail: meckel@informatik.uni-giessen.de
[4]E-mail: carlo.mereghetti@unimi.it
[5]E-mail: beatrice.palano@unimi.it

1 Introduction

Beside the formal definition of the accepted or generated language, the introduction of an accepting or generating device always brings the attention to several "auxiliary" formal structures related to the device itself. Such structures, which can be either crucial part of or derived from device definition, are not only interesting *per se*, but their investigation has often other relevant motivations. For instance, we can act on these structures to tune device computational capabilities. Or, they can directly imply certain device properties. Yet, they can also be used as a theoretical tool to get results in different contexts.

Only to cite some examples, in the realm of Turing machines, the well known language of valid computations is introduced in [11]. The study of this language has been widely used to point out undecidability of several problems and non-recursiveness of certain descriptional trade-offs (see, e.g., [7, 14]). In quantum automata theory, a basic part of the definition of several variants of quantum automata is the so-called control language which, very roughly speaking, describes computational dynamics which are admissible in a given model [19]. By modifying the control language, we obtain several quantum devices with different computational power. In [3], particular families of so-called selection languages are considered to tune the generative power of contextual grammars.

In this paper, we focus on *pushdown store languages* for pushdown automata (PDA). Given a PDA M, its pushdown store language $P(M)$ consists of all words occurring on the pushdown store along *accepting* computations of M. (It should be remarked that similar sets for stack automata have been investigated in [6].) It is known from [8] that, surprisingly enough, $P(M)$ is regular for any PDA. More recently, an alternative proof of this fact has been given in [1]. Here, we tackle the study of pushdown store languages from a descriptional complexity point of view, and design *succinct nondeterministic finite automata (NFA)* for their acceptance. In Section 3, a first general construction of an NFA for $P(M)$ resulting from [1] is presented. We subsequently improve this construction and obtain an upper bound to the size (i.e., number of states) of the NFA, which is quadratic in the number of states and linear in the number of pushdown symbols of the PDA M. Then, we show that this bound cannot be improved in general by showing its asymptotical optimality. In Section 4, we will be dealing with restricted versions of PDA, namely: PDA which cannot pop, stateless PDA, and counter machines, i.e., PDA whose pushdown store languages are subsets of Z^*Z_0, for a bottom-of-pushdown symbol Z_0 and a different pushdown symbol Z. For any of these restrictions, we present NFA for pushdown store languages which are strictly smaller than the NFA given for the general case. Moreover, in all cases, we prove the optimality of our constructions.

Finally, in Section 5, we apply our results on the descriptional complexity of pushdown store languages to the analysis of the hardness of some decision problems related to PDA. We show that the questions of whether $P(M)$ of a given PDA M is a finite set or is a finite set of words having at most length k, for a given $k \geq 1$, can be answered in deterministic polynomial time. Moreover, we also prove the P-completeness of these questions. As an application, we

obtain that it is P-complete to decide whether a given unambiguous PDA is a constant height PDA or is a PDA of constant height k, for a given $k \geq 1$ [2, 5]. As another application, we show the same complexity evaluation for the question of whether $P(M)$ is a subset of Z^*Z_0. This is equivalent to asking whether a given PDA is essentially a counter machine.

2 Preliminaries and Definitions

We assume that the reader is familiar with basic notions in formal language theory (see, e.g., [10, 15]). The set of natural numbers, with 0, is denoted by $\mathbb{N}$. The set of all words (including the empty word λ) over a finite alphabet Σ is denoted by Σ^*, and we let $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, and by $\Sigma^{\leq k}$ and $\Sigma^{>k}$ we denote, respectively, the set of all words of length less than or equal to k and larger than k. Given a language $L \subseteq \Sigma^*$, then $\mathit{suf}(L) = \{y \in \Sigma^* \mid \exists x \in \Sigma^*\colon xy \in L\}$ is the set of all suffixes of words in L. The reversal of a word w is denoted by w^R, and of a language L by L^R.

A *pushdown automaton* (PDA, see e.g. [10, 15]) is formally defined as a 7-tuple $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$, where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite pushdown alphabet, δ is the transition function mapping $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^*$, $q_0 \in Q$ is the initial state, $Z_0 \in \Gamma$ is a particular pushdown symbol, called the bottom-of-pushdown symbol, which initially appears on the pushdown store, and $F \subseteq Q$ is a set of accepting (or final) states. Roughly speaking, a *nondeterministic finite automaton* (NFA) can be viewed as a PDA where the pushdown store is never used.

A *configuration* of a pushdown automaton is a triple (q, w, γ), where q is the current state, w the unread part of the input, and γ the current content of the pushdown store, the *leftmost* symbol of γ being the *top* symbol. For $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $\gamma, \beta \in \Gamma^*$, and $Z \in \Gamma$, we write $(q, aw, Z\gamma) \vdash (p, w, \beta\gamma)$ whenever $(p, \beta) \in \delta(q, a, Z)$. As usual, the reflexive transitive closure of $\vdash$ is denoted by $\vdash^*$. The language accepted by M by *accepting states* is the set

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (f, \lambda, \gamma), \text{ for some } f \in F \text{ and } \gamma \in \Gamma^*\}.$$

Throughout the paper, unless otherwise stated, we will always be considering acceptance by accepting states. We measure the *size* of a PDA M by the product of the number of states, the number of pushdown symbols, the number of input symbols, and the maximum length $\mu(M)$ of strings of pushdown symbols appearing in the transition rules, i.e., $|Q| \cdot |\Gamma| \cdot |\Sigma| \cdot \mu(M)$. We also refer to [14] for a the discussion on PDA size measuring.

We consider PDA to be in *normal* form, i.e., with $\mu(M) \leq 2$. In this case, the pushdown height grows at most by 1 at each move. By introducing new states for every transition of a given PDA, it can be shown similarly to [14] that every PDA of size n can be converted to an equivalent normal form PDA of size $O(n)$. *For the rest of the paper, we assume that all PDA considered are in normal form.*

The *pushdown store language* of a PDA M (see, e.g., [1, 8]) is defined as the set $P(M)$ of all words occurring on the pushdown store along accepting computations of M. Formally:

$$P(M) = \{u \in \Gamma^* \mid \exists x, y \in \Sigma^*,\ q \in Q,\ f \in F : (q_0, xy, Z_0) \vdash^* (q, y, u) \vdash^* (f, \lambda, \gamma),\ \text{for some } \gamma \in \Gamma^*\}.$$

To clarify the notion of pushdown store language, we continue with an example.

Example 1. The language $\{ a^n b^n \mid n \geq 1 \}$ is accepted by the following (deterministic) PDA $M = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{Z, Z_0\}, \delta, q_0, Z_0, \{q_2\} \rangle$ such that

$$\delta(q_0, a, Z_0) = \{(q_0, ZZ_0)\},\ \delta(q_0, a, Z) = \{(q_0, ZZ)\},$$
$$\delta(q_0, b, Z) = \{(q_1, \lambda)\},\ \delta(q_1, b, Z) = \{(q_1, \lambda)\},\ \delta(q_1, \lambda, Z_0) = \{(q_2, Z_0)\}.$$

It is easy to see that the pushdown store language is $P(M) = Z^* Z_0$. □

3 Pushdown Store Languages: the General Case

Already in [8], it is proved that the pushdown store language $P(M)$ of a PDA M is regular. However, here we quickly review the proof given in [1], whose constructions enable us to obtain better bounds on the size of NFA accepting $P(M)$. In what follows, for $x \in \Sigma^*$, we use the short-hand notation

$$(p, w) \vdash^x (p', w') \text{ if and only if } (p, x, w) \vdash^* (p', \lambda, w').$$

Theorem 2 ([1]). *The pushdown store language of any PDA is regular.*

Proof. The construction given in [1] can be summarized as follows. Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ be a PDA. For every $q \in Q$, the following sets are defined:

$$Acc(q) = \{u \in \Gamma^* \mid \exists x, y \in \Sigma^* : (q_0, xy, Z_0) \vdash^* (q, y, u)\},$$
$$Co\text{-}Acc(q) = \{u \in \Gamma^* \mid \exists y \in \Sigma^*,\ f \in F,\ u' \in \Gamma^* : (q, y, u) \vdash^* (f, \lambda, u')\}.$$

Then, the pushdown store language is easily seen to be

$$P(M) = \bigcup_{q \in Q} Acc(q) \cap Co\text{-}Acc(q).$$

Now, for every $q \in Q$, we construct a left-linear grammar $G_{Acc(q)}$ for $Acc(q)$ and a right-linear grammar $G_{Co\text{-}Acc(q)}$ for $Co\text{-}Acc(q)$, thus showing that $P(M)$ is regular. Informally speaking, $G_{Acc(q)}$ simulates the behavior of M from an initial configuration until the state q is entered, while generating u on the pushdown store. Formally, $G_{Acc(q)}$ has terminal alphabet Γ, nonterminal alphabet $Q \times \Gamma$, start symbol (q_0, Z_0), and the following rules:

(1) $(p, Z) \longrightarrow (p', Z')$, if there exists $x \in \Sigma^*$ such that $(p, Z) \vdash^x (p', Z')$
(2) $(p, Z) \longrightarrow (p', Z')t$, if there exists $a \in \Sigma \cup \{\lambda\}$ with $(p', Z't) \in \delta(p, a, Z)$
(3) $(p, Z) \longrightarrow \lambda$, if there exists $x \in \Sigma^*$ such that $(p, Z) \vdash^x (q, \lambda)$
(4) $(q, Z) \longrightarrow Z$.

It is shown in [1] that $G_{Acc(q)}$ generates $Acc(q)$. On the other hand, $G_{Co\text{-}Acc(q)}$ simulates the behavior of M from configurations having q as state, u as pushdown content, and reaching an accepting state. Formally, $G_{Co\text{-}Acc(q)}$ has terminal alphabet Γ, nonterminal alphabet $Q \cup \{s\}$, start symbol q, and the following rules:

(5) $p \longrightarrow Zp'$, if there exists $x \in \Sigma^*$ such that $(p, Z) \vdash^x (p', \lambda)$
(6) $p \longrightarrow Zs$, if there exists $x \in \Sigma^*, p' \in F, u' \in \Gamma^+ : (p, Z) \vdash^x (p', u')$
(7) $s \longrightarrow Zs \mid \lambda$, for all $Z \in \Gamma$
(8) $p \longrightarrow \lambda$, if $p \in F$.

Again in [1], it is shown that $G_{Co\text{-}Acc(q)}$ generates $Co\text{-}Acc(q)$. Finally, it is important to stress that grammars $G_{Acc(q)}$ and $G_{Co\text{-}Acc(q)}$ can be effectively constructed. In particular, for establishing rules (1), (3), (5), and (6), the decidability of the emptiness problem for context-free languages is used (see, e.g., [15]). □

To clarify the meaning and construction of grammars $G_{Acc(q)}$ and $G_{Co\text{-}Acc(q)}$ given in the previous theorem, in the following example we provide such grammars for the PDA M in Example 1 accepting the language $\{a^n b^n \mid n \geq 1\}$.

Example 3. For the sake of brevity, we provide grammars only for $Acc(q_0)$ and $Co\text{-}Acc(q_0)$. However, the reader may easily verify that $P(M)$ consists of only $Acc(q_0) \cap Co\text{-}Acc(q_0)$. The grammar $G_{Acc(q_0)}$ (resp., $G_{Co\text{-}Acc(q_0)}$) has start symbol (q_0, Z_0) (resp., q_0), and the following rules:

$G_{Acc(q_0)}$	$G_{Co\text{-}Acc(q_0)}$
$(q_0, Z_0) \rightarrow (q_0, Z)Z_0$	$q_0 \rightarrow Zq_1$
$(q_0, Z) \rightarrow (q_0, Z)Z \mid Z$	$q_1 \rightarrow Zq_1 \mid Z_0 s$
	$s \rightarrow Zs \mid Z_0 s \mid \lambda$

It is not hard to verify that $G_{Acc(q_0)}$ generates $Z^* Z_0$, while $G_{Co\text{-}Acc(q_0)}$ gives $Z^* Z_0 \{Z, Z_0\}^*$. The intersection of these languages clearly yields $Z^* Z_0$, which is $P(M)$ as pointed out in Example 1. □

From the constructive proof of Theorem 2, we are able to provide a first upper bound on the size of NFA for pushdown store languages. From now on, for the descriptional costs of well known operations on NFA, the reader is referred to, e.g., [13, 21].

Proposition 4. *Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ be a PDA. Then, $P(M)$ can be accepted by an NFA with $|Q|^3|\Gamma| + |Q|^2(|\Gamma| + 1) + |Q| + 1$ many states.*

Proof. By Theorem 2, we have that $P(M) = \bigcup_{q \in Q} Acc(q) \cap Co\text{-}Acc(q)$. So, we begin by constructing, for any $q \in Q$, NFA accepting $Acc(q)$ and $Co\text{-}Acc(q)$. For $Acc(q)$, we start from the left-linear grammar $G_{Acc(q)}$ and construct a right-linear grammar generating the reversal $Acc(q)^R$ by simply interchanging every type (2) rule $(p, Z) \rightarrow (p', Z')t$ with $(p, Z) \rightarrow t(p', Z')$. Second, this right-linear grammar is directly converted to an equivalent NFA of size $|Q| \cdot |\Gamma|$ plus

an additional and unique accepting state due to type (3) and (4) rules. This gives an NFA N of size $|Q| \cdot |\Gamma| + 1$. Now, by basically "reverting arrows" and switching the initial and final state in N, we get an NFA $N_{Acc(q)}$ of the same size accepting $Acc(q)$. For $Co\text{-}Acc(q)$, the right-linear grammar $G_{Co\text{-}Acc(q)}$ can be directly translated into an equivalent NFA $N_{Co\text{-}Acc(q)}$ of size $|Q| + 1$.

From NFA $N_{Acc(q)}$, $N_{Co\text{-}Acc(q)}$, and by using well known constructions on NFA, we get an NFA with $(|Q| \cdot |\Gamma| + 1)(|Q| + 1)$ states for $Acc(q) \cap Co\text{-}Acc(q)$. Finally, the union over all $q \in Q$ can be implemented by an NFA of at most $|Q|(|Q| \cdot |\Gamma| + 1)(|Q| + 1) + 1 = |Q|^3|\Gamma| + |Q|^2(|\Gamma| + 1) + |Q| + 1$ states. □

We are now going to improve Proposition 4 and obtain a smaller NFA:

Theorem 5. *Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a PDA. Then, $P(M)$ is accepted by an NFA whose size is bounded by $|Q|^2(|\Gamma| + 1) + |Q|(2|\Gamma| + 3) + 2$.*

Proof. The key idea is to "get rid" of the union over $q \in Q$ in the definition of $P(M)$, by defining the set $Acc(Q)$ (resp., $Co\text{-}Acc(Q)$) representing all the pushdown contents reachable from the initial configuration (resp., from which a final state can be reached). Then, we have to build only the intersection of these two sets. More precisely, we let $[Q] = \{[q] \mid q \in Q\}$ and define the following sets:

$$Acc(Q) = \{[q]u \in [Q]\Gamma^* \mid u \in Acc(q)\},$$
$$Co\text{-}Acc(Q) = \{[q]u \in [Q]\Gamma^* \mid u \in Co\text{-}Acc(q)\}.$$

A left-linear grammar for $Acc(Q)$ is built similarly as $G_{Acc(q)}$ in Theorem 2, but we enlarge the terminal alphabet by the set $[Q]$, and have the following rules:

(1) $(p, Z) \longrightarrow (p', Z')$, if there exists $x \in \Sigma^*$ such that $(p, Z) \vdash^x (p', Z')$
(2) $(p, Z) \longrightarrow (p', Z')t$, if there exists $a \in \Sigma \cup \{\lambda\}$ such that $(p', Z't) \in \delta(p, a, z)$
(3)′ $(p, Z) \longrightarrow [q]$, if there exists $x \in \Sigma^*$ such that $(p, Z) \vdash^x (q, \lambda)$
(4)′ $(q, Z) \longrightarrow [q]Z$.

Similarly as in Proposition 4, from this left-linear grammar we can obtain an NFA for $Acc(Q)$ featuring $|Q| \cdot (|\Gamma| + 1) + 1$ many states.

The right-linear grammar for $Co\text{-}Acc(Q)$ is also built similarly as $G_{Co\text{-}Acc(q)}$ in Theorem 2. We enlarge again the terminal alphabet by $[Q]$, and we add a new nonterminal S which will be the start symbol. Then, we have the same rules (5) to (8) and additional rules $S \longrightarrow [p]p$, for all $p \in Q$. As in Proposition 4, this right-linear grammar can be converted to an NFA having at most $|Q| + 2$ states.

From the two obtained NFA, an NFA N for $Acc(Q) \cap Co\text{-}Acc(Q)$ can be constructed, featuring $(|Q| \cdot (|\Gamma| + 1) + 1)(|Q| + 2) = |Q|^2(|\Gamma| + 1) + |Q|(2|\Gamma| + 3) + 2$ many states. Now, notice that $P(M)$ can be obtained by deleting the initial symbols $[q]$ from every string in $L(N)$. Thus, by simply substituting in N the initial transitions taking place on $[q]$ with λ-transitions, we get an NFA for $P(M)$ with $|Q|^2(|\Gamma| + 1) + |Q|(2|\Gamma| + 3) + 2$ states. □

Our construction in Theorem 5 is the best possible. In fact, we are able to show an optimal lower bound of $\Omega(|Q|^2|\Gamma|)$. To achieve this, we start by introducing a simple language with the corresponding accepting PDA, and study the lower limit to the size of NFA for the pushdown store language of the given PDA. The result is contained in the following lemma:

Lemma 6. *For a fixed $m \geq 2$, let the single word language $L_m = \{a^{m^2}b\}$. Then, L_m can be accepted by a PDA $M_{m,1}$ of size $O(m)$ such that every NFA accepting $P(M_{m,1})$ needs at least $m^2 + 3$ states.*

Proof. Consider the PDA $M_{m,1} = \langle Q, \{a, b\}, \{Z_0', Z_0, Z_1\}, \delta, q_0, Z_0', \{q_{2m+1}\}\rangle$, where $Q = \{q_0, q_1, \ldots, q_{2m+1}\}$ and δ is defined by

$$\begin{aligned}
\delta(q_0, \lambda, Z_0') &= \{(q_0, Z_0 Z_0')\}, \\
\delta(q_0, a, Z_0) &= \{(q_1, Z_1 Z_0)\}, \\
\delta(q_i, a, Z_1) &= \{(q_{i+1}, Z_1 Z_1)\}, && \text{for } 1 \leq i \leq m-2, \\
\delta(q_{m-1}, a, Z_1) &= \{(q_0, Z_0 Z_1)\}, \\
\delta(q_0, b, Z_0) &= \{(q_m, \lambda)\}, \\
\delta(q_i, \lambda, Z_1) &= \{(q_i, \lambda)\}, && \text{for } m \leq i \leq 2m-1, \\
\delta(q_i, \lambda, Z_0) &= \{(q_{i+1}, \lambda)\}, && \text{for } m \leq i \leq 2m-1, \\
\delta(q_{2m}, \lambda, Z_0') &= \{(q_{2m+1}, Z_0')\}.
\end{aligned}$$

The PDA accepts by entering the accepting state q_{2m+1}. To determine $L(M_{m,1})$, we observe that every a of the input is encoded and written on the pushdown: every mth a is encoded as Z_0, while the remaining a's as Z_1. A transition processing b is possible only if in the input there has been a number of a's which is a multiple of m. In this case, the automaton switches to state q_m. When entering q_m for the first time, the topmost symbol is Z_1 and the pushdown contains, over the bottom-of-pushdown symbol Z_0', occurrences of the block $Z_1^{m-1}Z_0$. By λ-transitions, the state q_{2m} is only reached after deleting exactly m blocks $Z_1^{m-1}Z_0$. At this point, from q_{2m} and by having Z_0' on the pushdown, $M_{m,1}$ switches to the accepting state q_{2m+1}. Thus, we obtain that $L(M_{m,1}) = L_m$.

This reasoning gives the structure of the pushdown store language $P(M_{m,1})$. If the input is accepted, the longest content of the pushdown store consists of the symbol Z_0 on top, followed by exactly m blocks $Z_1^{m-1}Z_0$, plus the initial pushdown symbol Z_0', i.e., $Z_0(Z_1^{m-1}Z_0)^m Z_0'$. Since at every step only one symbol is added on the pushdown or deleted from it, every suffix of this word also belongs to the pushdown store language. Moreover, in all computations the pushdown is never empty, so λ does not belong to $P(M_{m,1})$. Thus, we conclude that $P(M_{m,1}) = \mathit{suf}(Z_0(Z_1^{m-1}Z_0)^m Z_0') \setminus \{\lambda\}$.

An NFA accepting $P(M_{m,1})$ needs exactly m^2+3 states since this language is finite and its longest word has length $m^2 + 2$. □

This result is then generalized to get the desired lower bound of $\Omega(|Q|^2|\Gamma|)$. More precisely, we consider PDA for the finite languages $L_{m,k} = \{(a^{m^2}b^{m^2})^{k/2}c\}$ for even k, and $L_{m,k} = \{(a^{m^2}b^{m^2})^{(k-1)/2}a^{m^2}c\}$ for odd $k \geq 3$, and prove lower limits to the size of NFA for their pushdown store languages. Results are contained in the following lemma:

Lemma 7. *For $m, k \geq 2$, there exist PDA $M_{m,k} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ accepting $L_{m,k}$ such that $|Q| = 2m + 2$ and $|\Gamma| = 2k + 1$. Moreover, any NFA for $P(M_{m,k})$ needs at least $k \cdot m^2 + 3 \in \Omega(|Q|^2|\Gamma|)$ states.*

Proof. For $m, k \geq 2$, let the PDA $M_{m,k} = \langle Q, \{a, b, c\}, \Gamma, \delta, q_0, Z'_0, \{q_{2m+1}\}\rangle$, where $Q = \{q_0, q_1, \ldots, q_{2m+1}\}$, $\Gamma = \{Z'_0, Z_0, Z_1, \ldots, Z_{2k-1}\}$, and δ is defined by

$$
\begin{aligned}
\delta(q_0, \lambda, Z'_0) &= \{(q_0, Z_0 Z'_0)\},\\
\delta(q_0, a, Z_i) &= \{(q_1, Z_{i+1} Z_i)\}, \text{ for } i < 2k-1, i \bmod 4 \equiv 0,\\
\delta(q_0, b, Z_i) &= \{(q_1, Z_{i+1} Z_i)\}, \text{ for } i < 2k-1, i \bmod 4 \equiv 2,\\
\delta(q_j, a, Z_i) &= \{(q_{j+1}, Z_i Z_i)\}, \text{ for } 1 \leq j \leq m-2, i \leq 2k-1, i \bmod 4 \equiv 1,\\
\delta(q_j, b, Z_i) &= \{(q_{j+1}, Z_i Z_i)\}, \text{ for } 1 \leq j \leq m-2, i \leq 2k-1, i \bmod 4 \equiv 3,\\
\delta(q_{m-1}, a, Z_i) &= \begin{cases} \{(q_0, Z_{i-1} Z_i), (q_0, Z_{i+1} Z_i)\}, & \text{if } i < 2k-1, i \bmod 4 \equiv 1,\\ \{(q_0, Z_{i-1} Z_i)\}, & \text{if } i = 2k-1, \end{cases}\\
\delta(q_{m-1}, b, Z_i) &= \begin{cases} \{(q_0, Z_{i-1} Z_i), (q_0, Z_{i+1} Z_i)\}, & \text{if } i < 2k-1, i \bmod 4 \equiv 3,\\ \{(q_0, Z_{i-1} Z_i)\}, & \text{if } i = 2k-1, \end{cases}\\
\delta(q_0, c, Z_{2k-2}) &= \{(q_m, \lambda)\},\\
\delta(q_j, \lambda, Z_{2i+1}) &= \{(q_j, \lambda)\}, \quad \text{for } 0 \leq i \leq k-1, m \leq j \leq 2m-1,\\
\delta(q_j, \lambda, Z_{2i}) &= \{(q_{j+1}, \lambda)\}, \text{ for } 0 \leq i \leq k-1, m \leq j \leq 2m-2,\\
\delta(q_{2m-1}, \lambda, Z_{2i}) &= \begin{cases} \{(q_m, \lambda)\}, & \text{if } 1 \leq i \leq k-1,\\ \{(q_{2m}, \lambda)\}, & \text{if } i = 0, \end{cases}\\
\delta(q_{2m}, \lambda, Z'_0) &= \{(q_{2m+1}, Z'_0)\}.
\end{aligned}
$$

This automaton accepts a word when ending up in the accepting state q_{2m+1}. To determine $L(M_{m,k})$, we can observe that every a and b of the input is encoded and written on the pushdown: every mth symbol is encoded by a pushdown symbol with even index and the remaining ones by one with an odd index. A block of a's needs to be followed by a block of b's and vice versa; c follows the last block. A transition by c is only possible if there have been exactly k blocks of a's and b's in the input, each block with length multiple of m. In this case, the automaton switches to state q_m.

When entering state q_m for the first time, the topmost symbol is Z_{2k-1} and the pushdown contains occurrences of blocks of the form $Z_{2i+1}^{m-1} Z_{2i}$ on top of the bottom-of-pushdown symbol Z'_0. By λ-transitions, the state q_{2m} is only reached after deleting exactly m blocks $Z_{2i+1}^{m-1} Z_{2i}$, for $k-1 \geq i \geq 0$ in descending order. At this point, from q_{2m} and by having Z'_0 on the pushdown, $M_{m,k}$ switches to the accepting state q_{2m+1}. So, $L(M_{m,k}) = \{(a^{m^2} b^{m^2})^{k/2} c\}$ if k is even, and $L(M_{m,k}) = \{(a^{m^2} b^{m^2})^{(k-1)/2} a^{m^2} c\}$ if k is odd.

The considerations above give the structure of the pushdown store language $P(M_{m,k})$. If the input is accepted, the longest content of the pushdown store consists of the symbol Z_{2k-2} on top, followed by exactly m blocks of $Z_{2i+1}^{m-1} Z_{2i}$, for $k-1 \geq i \geq 0$, plus the initial pushdown symbol Z'_0, i.e.:

$$Z_{2k-2} (Z_{2k-1}^{m-1} Z_{2k-2})^m (Z_{2k-3}^{m-1} Z_{2k-4})^m \cdots (Z_3^{m-1} Z_2)^m (Z_1^{m-1} Z_0)^m Z'_0.$$

Again, every suffix of this word, except λ, belongs to $P(M_{m,k})$, so

$$P(M_{m,k}) = \mathit{suf}(Z_{2k-2} (Z_{2k-1}^{m-1} Z_{2k-2})^m (Z_{2k-3}^{m-1} Z_{2k-4})^m \cdots (Z_1^{m-1} Z_0)^m Z'_0) \setminus \{\lambda\}.$$

An accepting NFA for $P(M_{m,k})$ needs exactly $k{\cdot}m^2{+}3$ states, since this language is finite and its longest word has length $k \cdot m^2 + 2$. □

As a consequence of Theorem 5 and Lemma 7, we obtain that our construction of NFA for pushdown store languages is the best possible:

Theorem 8. *Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a PDA. Then, an NFA for $P(M)$ exists with $O(|Q|^2|\Gamma|)$ states. On the other hand, there exist infinitely many PDA $M_{Q,\Gamma}$ of size $O(|Q| \cdot |\Gamma|)$ such that every NFA accepting $P(M_{Q,\Gamma})$ needs $\Omega(|Q|^2|\Gamma|)$ states.*

4 Pushdown Store Languages for Special Cases

Here, we consider restricted models of PDA for which we are able to provide NFA for their pushdown store languages, whose size is strictly below the general upper bound given in Theorem 5.

4.1 PDA which can never pop

As a first restriction, we consider PDA *which never pop* a symbol from the pushdown. Thus, for such devices, once stored in the pushdown, symbols can never be modified. It is easy to see that these PDA accept exactly the regular languages. In what follows, given a PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ of this kind, we quickly outline the construction of an NFA for $P(M)$ of size $|Q| \cdot |\Gamma| + 1$.

It is not hard to see that, in this case, we have

$$P(M) = \{u \in \Gamma^* \mid u \in Acc(q) \text{ and } q \in F\}.$$

Moreover, the left-linear grammars for $Acc(q)$, with $q \in F$, given in Theorem 2 now drop type (1) and type (3) rules, and can be summarized in a single left-linear grammar with terminal alphabet Γ, nonterminal alphabet $Q \times \Gamma$, start symbol (q_0, Z_0), and the following rules:

(1) $(p, Z) \longrightarrow (p', Z')t$, if there exists $a \in \Sigma \cup \{\lambda\}$ such that $(p', Z't) \in \delta(p, a, Z)$
(2) $(q, Z) \longrightarrow Z$, if $q \in F$.

From this left-linear grammar, we can easily obtain an NFA for $P(M)$ with $|Q| \cdot |\Gamma| + 1$ states. This upper bound is also tight, as shown in the following lemma:

Lemma 9. *For $m, k \geq 2$, there exist PDA $M_{m,k}$ which can never pop having m states and k pushdown symbols, for which every NFA for $P(M_{m,k})$ needs at least $k \cdot m + 1$ states.*

Proof. For $m, k \geq 2$, let the PDA $M = \langle Q, \{a, b\}, \Gamma, \delta, q_0, Z_0, \{q_{m-1}\}\rangle$, where $Q = \{q_0, q_1, \ldots, q_{m-1}\}$, $\Gamma = \{Z_0, Z_1, \ldots, Z_{k-1}\}$, and δ is defined as follows.

For even k and by letting $X = \{0, 2, 4, \ldots, k-2\}$, we set

$$\begin{aligned}
\delta(q_j, a, Z_i) &= \{(q_j, Z_i Z_i)\}, && \text{for } i \in X, 0 \leq j \leq m-1,\\
\delta(q_j, b, Z_{i+1}) &= \{(q_j, Z_{i+1} Z_{i+1})\}, && \text{for } i \in X, 0 \leq j \leq m-1,\\
\delta(q_j, a, Z_{i+1}) &= \{(q_j, Z_{i+2} Z_{i+1})\}, && \text{for } i \in X \setminus \{k-2\}, 0 \leq j \leq m-1,\\
\delta(q_j, b, Z_i) &= \{(q_j, Z_{i+1} Z_i)\}, && \text{for } i \in X, 0 \leq j \leq m-1,\\
\delta(q_j, a, Z_{k-1}) &= \{(q_{j+1}, Z_0 Z_{k-1})\}, && \text{for } 0 \leq j \leq m-2.
\end{aligned}$$

For odd k and by letting $Y = \{0, 2, 4, \ldots, k-1\}$, we define δ as

$$\begin{aligned}
\delta(q_j, a, Z_i) &= \{(q_j, Z_i Z_i)\}, && \text{for } i \in Y, 0 \leq j \leq m-1, j \text{ is even},\\
\delta(q_j, b, Z_{i+1}) &= \{(q_j, Z_{i+1} Z_{i+1})\}, && \text{for } i \in Y \setminus \{k-1\}, 0 \leq j \leq m-1, j \text{ is even},\\
\delta(q_j, a, Z_{i+1}) &= \{(q_j, Z_{i+2} Z_{i+1})\}, && \text{for } i \in Y \setminus \{k-1\}, 0 \leq j \leq m-1, j \text{ is even},\\
\delta(q_j, b, Z_i) &= \{(q_j, Z_{i+1} Z_i)\}, && \text{for } i \in Y \setminus \{k-1\}, 0 \leq j \leq m-1, j \text{ is even},\\
\delta(q_j, b, Z_{k-1}) &= \{(q_{j+1}, Z_0 Z_{k-1})\}, && \text{for } 0 \leq j \leq m-2, j \text{ is even},\\
\delta(q_j, b, Z_i) &= \{(q_j, Z_i Z_i)\}, && \text{for } i \in Y, 0 \leq j \leq m-1, j \text{ is odd},\\
\delta(q_j, a, Z_{i+1}) &= \{(q_j, Z_{i+1} Z_{i+1})\}, && \text{for } i \in Y \setminus \{k-1\}, 0 \leq j \leq m-1, j \text{ is odd},\\
\delta(q_j, b, Z_{i+1}) &= \{(q_j, Z_{i+2} Z_{i+1})\}, && \text{for } i \in Y \setminus \{k-1\}, 0 \leq j \leq m-1, j \text{ is odd},\\
\delta(q_j, a, Z_i) &= \{(q_j, Z_{i+1} Z_i)\}, && \text{for } i \in Y \setminus \{k-1\}, 0 \leq j \leq m-1, j \text{ is odd},\\
\delta(q_j, a, Z_{k-1}) &= \{(q_{j+1}, Z_0 Z_{k-1})\}, && \text{for } 0 \leq j \leq m-2, j \text{ is odd}.
\end{aligned}$$

Notice that M never pops a symbol from the pushdown, and accepts a word when ending up in the accepting state q_{m-1}. To give an idea of the dynamics of M, we quickly describe the behavior on an input $w \in L(M)$. Every symbol of w is encoded and written on the pushdown. So, the pushdown store contains $|w|+1$ symbols at the end of an accepting computation. For an even k, transitions guarantee that w consists of a concatenation of arbitrarily long a-blocks and b-blocks, which starts with a (possibly empty) a-block, and then continues with an alternation of non-empty b-blocks and non-empty a-blocks. There is only one exception at the end of the string since, whenever M enters the state q_{m-1}, the input could be accepted. Moreover, the number of b-blocks is $\frac{k}{2} \cdot (m-1)$ or $\frac{k}{2} \cdot m$, while the number of a-blocks is at least $\frac{k}{2} \cdot (m-1) - 1$ and at most $\frac{k}{2} \cdot m$. All this reasoning leads us to formally settle $L(M)$. We define the set *pre* of a regular expression or language as the set containing all prefixes of words of the given language. So, we get that $L(M) = ((a^*bb^*a)^{k/2})^{m-1} pre((a^*bb^*a)^{k/2})$ for even k.

We leave to the reader to work out the details for the construction of $L(M)$ for odd k; here, we give its final form. For odd k and odd m, we have

$$\begin{aligned}
L(M) = (a^*bb^*a)^{(k-1)/2} a^* ((bb^*aa^*)^{(k-1)/2} bb^* (aa^*bb^*)^{(k-1)/2} aa^*)^{(m-1)/2-1}\\
(bb^*aa^*)^{(k-1)/2} bb^* pre((aa^*bb^*)^{(k-1)/2} aa^*),
\end{aligned}$$

while for odd k and even m, we have

$$\begin{aligned}
L(M) = (a^*bb^*a)^{(k-1)/2} a^* ((bb^*aa^*)^{(k-1)/2} bb^* (aa^*bb^*)^{(k-1)/2} aa^*)^{m/2-1}\\
pre((bb^*aa^*)^{(k-1)/2} bb^*).
\end{aligned}$$

The pushdown store language $P(M)$ can be entailed by examining the encoding in the pushdown store of accepted input words. on The first k blocks are encoded

by Z_0 to Z_{k-1} if the word begins with an a. Otherwise, there exist only $k-1$ blocks that are encoded by Z_1 to Z_{k-1}. For the next k blocks the first one starts with an a. So, this block is encoded by Z_0 to Z_{k-1}. This is repeated for the next k blocks and so on. Since the pushdown is never empty, the empty word λ does not belong to $P(M)$. We obtain the language

$$P(M) = \mathit{suf}((Z_{k-1}Z_{k-1}^* \cdots Z_iZ_i^* \cdots Z_1Z_1^*Z_0^*Z_0)^m) \setminus \{\lambda\}.$$

Notice that $P(M)$ contains the word $(Z_{k-1} \cdots Z_2Z_1Z_0)^m$ whose length is $k \cdot m$, and that all the other words in $P(M)$ may be accepted on an NFA by "elaborating" on an accepting computation on this word; details may be easily filled. Then, it is not hard to see that $k \cdot m+1$ states are necessary and sufficient for an NFA to accept $P(M)$. □

Altogether, for PDA which can never pop, we get that $|Q| \cdot |\Gamma| + 1$ is a tight bound for the size of NFA accepting their pushdown store languages.

4.2 Stateless PDA

Next, we consider PDA which have *one state only*, whose acceptance policy is clearly by *empty pushdown* (i.e., they accept by completely sweeping the input and emptying the pushdown store [10, 15]). Such devices are also called *stateless PDA*. It is known (see, e.g., [10]) that their deterministic version characterizes the class of *simple* languages. Clearly, for stateless PDA the upper bound of Theorem 5 reduces to $3 \cdot |\Gamma| + 6$. However, we are now going to provide a better upper bound.

Let us apply the constructions in the proof of Theorem 2 to a stateless PDA $M = \langle \{q_0\}, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset \rangle$. For $Acc(q_0)$, we can take $\langle \Gamma \rangle = \{\langle Z \rangle \mid Z \in \Gamma\}$ as set of nonterminal symbols, and consider rules of the following form:

(1) $\langle Z \rangle \longrightarrow \langle Z' \rangle$, if there exists $x \in \Sigma^*$ such that $(q_0, Z) \vdash^x (q_0, Z')$
(2) $\langle Z \rangle \longrightarrow \langle Z' \rangle t$, if there exists $a \in \Sigma \cup \{\lambda\}$ such that $(q_0, Z't) \in \delta(q_0, a, Z)$
(3) $\langle Z \rangle \longrightarrow \lambda$, if there exists $x \in \Sigma^*$ such that $(q_0, Z) \vdash^x (q_0, \lambda)$
(4) $\langle Z \rangle \longrightarrow Z$.

This leads to an NFA of size at most $|\Gamma| + 1$. For the set $\mathit{Co\text{-}Acc}(q_0)$, we only have to consider rules of the form

(5) $q_0 \longrightarrow Zq_0$, if there exists $x \in \Sigma^*$ such that $(q_0, Z) \vdash^x (q_0, \lambda)$
(6) $q_0 \longrightarrow \lambda$.

An equivalent NFA can be trivially constructed, with exactly one state. Thus, an NFA for $P(M) = Acc(q_0) \cap \mathit{Co\text{-}Acc}(q_0)$ needs at most $|\Gamma| + 1$ states. The optimality of this bound is shown as follows:

For $k \geq 0$, we consider the following family of stateless (deterministic) PDA $M_k = \langle \{q_0\}, \{a, b\}, \{Z_0, A_1, A_2, \ldots, A_k\}, \delta, q_0, Z_0, \emptyset \rangle$ with

$$\begin{aligned}
&\delta(q_0, a, Z_0) = \{(q_0, A_1Z_0)\} \\
&\delta(q_0, a, A_i) = \{(q_0, A_{i+1}A_i)\}, \text{ for } 1 \leq i < k \\
&\delta(q_0, b, Z_0) = \{(q_0, \lambda)\} \\
&\delta(q_0, b, A_i) = \{(q_0, \lambda)\}, \qquad \text{for } 1 \leq i \leq k.
\end{aligned}$$

We observe that $L(M_k) = \{a^i b^{i+1} \mid i \leq k\}$. Hence, it is not difficult to see that

$$P(M_k) = \{\lambda, Z_0, A_1 Z_0, A_2 A_1 Z_0, \ldots, A_k \cdots A_2 A_1 Z_0\}.$$

Clearly, for any $k \geq 0$, we have that any NFA accepting $P(M_k)$ needs exactly $k+2$ states, whereas the size of the pushdown alphabet is $k+1$.

Altogether, we get the following

Theorem 10. *Let $M = \langle\{q_0\}, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset\rangle$ be a stateless PDA. Then, every NFA for $P(M)$ needs at most $|\Gamma|+1$ states. Moreover, for any $k \geq 0$, there exists a stateless PDA M_k having $|\Gamma_k| = k+1$ pushdown symbols, for which every NFA for $P(M_k)$ needs at least $k+2 = |\Gamma_k|+1$ states.*

4.3 Counter machines

A *counter machine* (see, e.g., [9, 15]) is defined as a traditional PDA with the restriction that the pushdown alphabet contains only two symbols, namely, the usual bottom-of-pushdown symbol Z_0, which can never be pushed, and a distinguished symbol Z. For the sake of simplicity, we also assume that Z_0 can never be popped. This assumption does not affect our results at all, but avoids a lot of tedious technicalities. So, the pushdown store language of a counter machine is a subset of Z^*Z_0 (see also [12]). As an example, the PDA in Example 1 is actually a counter machine.

We are going to show that the size of NFA accepting the pushdown store language of counter machines is linear in $|Q|$ and not quadratic as proved in Theorem 5 for general PDA. Moreover, we will show the optimality of such a size bound. Let us begin by observing the following easy fact:

Lemma 11. *Given a counter machine M, then $P(M)$ is either Z^*Z_0 or $Z^{\leq h}Z_0$ for a fixed $h \geq 0$.*

For a counter machine having pushdown store language of the form $Z^{\leq h}Z_0$, we are able to bound h by the number of finite control states:

Theorem 12. *Given the counter machine $M = \langle Q, \Sigma, \{Z_0, Z\}, \delta, q_0, Z_0, F\rangle$, let $P(M) = Z^{\leq h}Z_0$ for a fixed $h \geq 0$. Then, $h \leq |Q|$.*

Proof. Let us assume that $h > |Q|$. We are going to prove that $P(M) = Z^*Z_0$, thus contradicting the supposed finiteness of $P(M)$. Consider an accepting computation Π of M on an input string $x \in L(M)$, along which the pushdown content Z^hZ_0 appears.

To deal with the "hardest" situation, we assume that Z^hZ_0 appears during a *branch* of Π at the end of which the pushdown store is emptied. So, more formally, Π has a branch of computation with the following features:

(1) it starts from a configuration with pushdown content ZZ_0,
(2) it reaches a configuration where the pushdown content Z^hZ_0 shows up,
(3) it starts from such a configuration and ends in a configuration with pushdown content ZZ_0,

(4) it never finds itself in a configuration where the pushdown contains Z_0 only.

At the end of this proof, we will fix the case in which (3) does not hold within Π. We call *ascending (resp., descending)* the sub-branch between breakpoints (1) and (2) (resp., (2) and (3)). During the ascending sub-branch, M being in normal form, there must exist a sequence of configurations having the form $\{(q^{(i)}, x^{(i)}, Z^i Z_0)\}_{i=1}^{h}$. Since $h > |Q|$, there exist $1 \leq r < s \leq h$ such that $q^{(r)} = q^{(s)} = q$ and $x^{(r)} = wx^{(s)}$, for some $w \in \Sigma^*$. It is easy to see that, upon consuming the substring w, M increases the pushdown content by the factor Z^t with $0 < t = s - r$.

For the descending sub-branch, again due to the normal form, we get the existence of a sequence of configurations of the form $\{(p^{(i)}, y^{(i)}, Z^{h-i} Z_0)\}_{i=0}^{h-1}$. Since $h > |Q|$, there exist $0 \leq r' < s' \leq h-1$ such that $p^{(r')} = p^{(s')} = p$ and $y^{(r')} = w' y^{(s')}$, for some $w' \in \Sigma^*$. Now, upon consuming w', M deletes the factor $Z^{t'}$ from the pushdown store, with $0 < t' = s' - r'$.

All this reasoning enables us to factorize the input string as $x = uwzw'v$, on which the accepting computation Π presents the following breakpoints:

$$(q_0, uwzw'v, Z_0) \vdash^* (q, wzw'v, Z^r Z_0) \vdash^* (q, zw'v, Z^{r+t} Z_0) \vdash^*$$
$$\vdash^* (p, w'v, Z^{h-r'} Z_0) \vdash^* (p, v, Z^{h-r'-t'} Z_0) \vdash^* (f, \lambda, \gamma), \quad \text{with } f \in F,\ \gamma \in \Gamma^*.$$

Fig. 1. Breakpoints in the accepting computation on x.

For reader's ease of mind, we assume that both w and w' are not the empty string; however, situations in which this does not hold can be easily dealt with. Now, the idea is to construct from x a family of strings in $L(M)$ by suitably pumping factors w and w', whose accepting computations, "originating" from Π, will display all the strings $Z^* Z_0$ as pushdown contents. By starting from the original accepting computation Π of M on x, we obtain accepting computations for strings $uw^{\alpha+1} zw'^{\beta+1} v$, with $\alpha = t'\gamma$, $\beta = t\gamma$, and any $\gamma > 0$, as follows (the reader may follow the birth of these accepting computations by scanning Fig. 1):

- PROCESSING $uw^{\alpha+1}$: After reading u, M is in the state q and has $Z^r Z_0$ as pushdown content. Processing the first occurrence of w leaves M in q again with pushdown content $Z^{r+t} Z_0$. Notice that, along this processing, Z is always seen on top of the pushdown due to feature (4). Clearly, we can repeat such processing α times upon consuming the subsequent factor w^α, and finally reach the state q with pushdown content $Z^{s+\alpha t} Z_0$.
- PROCESSING z: This branch of computation takes place exactly as in Π, but starting with the pushdown store at a higher level. Precisely, we start from the state q with pushdown content $Z^{s+\alpha t} Z_0$, and end in p with pushdown content $Z^{h-r'+\alpha t} Z_0$.
- PROCESSING $w'^{\beta+1}$: After reading the first occurrence of w', M is again in the state p with pushdown content $Z^{h-r'+\alpha t-t'} Z_0$. We can repeat this branch of computation on the subsequent factor w'^β provided that we always have Z on top of the pushdown, which is always the case. Indeed, upon consuming w'^β, we delete a number of Z symbols which is $\beta t' = t\gamma t' = \alpha t$, thus reducing the pushdown content to $Z^{h-r'-t'} Z_0 = Z^{h-s'} Z_0$. Notice that, at this point, M is again in the state p.

– PROCESSING v: This branch of computation takes place exactly as in Π, starting from the configuration $(p, v, Z^{h-s'}Z_0)$ and ending in a final state.

The reader may easily verify that along the above constructed computation, the highest pushdown content is $Z^{h+t\gamma t'}Z_0$, reached at the end of the ascending sub-branch. Since this reasoning holds for any $\gamma > 0$, we get that there cannot exist any given constant bounding the pushdown height of M along every accepting computation. This clearly implies that $P(M) = Z^*Z_0$.

As a final fixing, we discuss the case in which feature (3) does not hold within Π. In this case, there exists a branch of computation in Π which: (i) starts from a configuration with ZZ_0 as pushdown content, (ii) reaches a configuration with Z^hZ_0 as pushdown content, but then (iii) never empties the pushdown. The sub-branch from (i) to (ii) again guarantees the existence of a factor w in x upon which M starts and end in the same state q, adding on the pushdown a factor Z^t, for $t > 0$. So, we can "pump the computation" of M on this factor and obtain accepting computations on strings $uw^\gamma v$ with pushdown height $h + \gamma t$. Indeed, after reading the last occurrence of w, M is in the state q and, while processing v, the symbol Z is always seen as top of the pushdown. So, M will end up in a final state, as in the original computation Π for x. □

As a consequence of Theorem 12, we are able to provide the exact size cost of accepting pushdown store languages of counter machines by NFA:

Theorem 13. *Let M be a counter machine with state set Q. Then, $P(M)$ is accepted by some NFA with size bounded by $|Q| + 2$. Moreover, this size bound is optimal.*

Proof. By Lemma 11 and Theorem 12, we learn that either $P(M) = Z^*Z_0$ or $P(M) = Z^{\leq h}Z_0$ with $h \leq |Q|$. In the former case, a 2-state NFA exists and the upper bound is clearly satisfied. In the latter, $P(M)$ is accepted by an NFA with $(|Q| + 2)$ states.

Concerning the optimality, we consider the two words language $L = \{\lambda, a^m\}$, for a fixed $m > 0$. It is not hard to design a (deterministic) counter machine M which accepts L with m states. Basically, M has states $q_0, q_1, \ldots, q_{m-1}$, where q_0 is both the initial and unique final state, and transition function

$$\begin{aligned} \delta(q_0, a, Z_0) &= \{(q_1, ZZ_0)\} \\ \delta(q_i, a, Z) &= \{(q_{i+1}, ZZ)\}, \text{ for } 1 \leq i \leq m-2 \\ \delta(q_{m-1}, a, Z) &= \{(q_0, ZZ)\}. \end{aligned}$$

The reader may easily verify that: on strings of length less than m, M rejects by ending in a non-accepting state, on strings of length greater than m, M rejects by being blocked in q_0 after consuming m symbols. would be to have Z_0) On the other hand, processing a^m exactly leaves M in q_0 after completely sweeping the input, so we accept.

At this point, we only need to observe that $m + 2$ states are necessary and sufficient to accept $P(M) = Z^{\leq m}Z_0$ by a NFA. □

5 Computational Complexity of Decidability Questions

In this section, we study the complexity of deciding some properties of $P(M)$ for a given PDA M, namely, finiteness and being subset of Z^*Z_0. These two questions can be answered by constructing the NFA N for $P(M)$ and then deciding, respectively, finiteness or inclusion in Z^*Z_0 for $L(N)$. We show that the construction of N as well as the decision of both questions can be done in deterministic polynomial time with respect to the size of M. On the other hand, we also prove the P-completeness of these two decision problems. As a consequence, we get the P-completeness of deciding whether a PDA is of a certain "nature." For a background on complexity theory, we refer to, e.g., [4, 15].

Lemma 14. *Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a PDA. Then, an NFA for $P(M)$ can be constructed in deterministic polynomial time.*

Proof. We will show that the construction given in Theorem 5 can be done in polynomial time with respect to the size of M. Let us first take a close look at the construction of the set $Acc(Q)$. We have to consider every pair $(p, Z) \in Q \times \Gamma$ and, in particular, to test the existence of $x \in \Sigma^*$ such that $(p, Z) \vdash^x (p', Z')$ for some other pair $(p', Z') \in Q \times (\Gamma \cup \{\lambda\})$. As recalled in the proof of Theorem 2, this test is in essence an instance of the emptiness problem for context-free languages, which is known to be decidable in polynomial time. Since there are at most $|Q|^2(|\Gamma| + 1)^2$ such pairs, which is polynomial in the size of M, we obtain that the NFA for $Acc(Q)$ can be constructed in polynomial time. A similar reasoning holds for the NFA for $Co\text{-}Acc(Q)$. At this point, the NFA for $Acc(Q) \cap Co\text{-}Acc(Q)$ plus deleting the first symbol of every word in this intersection can be done in polynomial time as well. □

This is the first step to prove the P-completeness of the above mention decision problems on pushdown store languages:

Theorem 15. *Given a PDA M, it is* P*-complete to decide whether: (i) $P(M)$ is a finite set. (ii) $P(M)$ is a finite set of words having at most length k, for a given $k \geq 1$.*

Proof. Let us first show (i). The problem belongs to P: by Lemma 14, an NFA N for $P(M)$ can be built in polynomial time. Then, the infiniteness of $L(N)$ can be decided in NLOGSPACE [17]. Since NLOGSPACE is closed under complementation [16, 20], we get that the finiteness of $L(N)$ can be decided in NLOGSPACE $\subseteq$ P as well.

To show the completeness of the problem, we log-space reduce to it the emptiness problem for context-free grammars, which is known to be P-complete due to [18]. Given a context-free grammar $G = \langle N, T, S, R\rangle$, let $\$ \notin T$ be a new terminal symbol, and $S', S'' \notin N$ be new nonterminals. We define the context-free grammar

$$G' = \langle N \cup \{S', S''\}, T \cup \{\$\}, S', R \cup \{S' \to SS'', S'' \to S''\$ \mid \$\}\rangle.$$

Observe that $L(G') = L(G)\$^+$. From G', we construct an equivalent PDA M' using the standard construction [15], where a stateless PDA is built which mimics a left derivation of G'. More precisely, let $M' = \langle\{q\}, T \cup \{\$\}, N \cup T \cup \{S', S'', \$\}, \delta, q, S', \emptyset\rangle$. Observe that G' is not in Greibach normal form, but this is not essential for the construction in [15] since we can define δ to have the following transitions: $(q, \gamma) \in \delta(q, \lambda, A)$, if $A \to \gamma$ belongs to G' for all $A \in N \cup \{S', S''\}$ and $\gamma \in (N \cup T \cup \{S', S'', \$\})^*$, and $(q, \lambda) \in \delta(q, a, a)$ for all $a \in T \cup \{\$\}$. Moreover, M' can be converted to an equivalent PDA which is in normal form and accepts by accepting states. This conversion increases the size of M' at most linearly. Due to the left-recursive rule $S'' \to S''\$$, we obtain that $P(M')$ *is finite if and only if* $L(G)$ *is empty.*

To conclude the proof of (i), we have to make sure that our reduction can be done in deterministic logarithmic space. We quickly notice that the only possibly costly operation is the construction from [15] for the PDA M'. However, this construction gives that M' has one state, $|N| + |T| + 3$ pushdown symbols, and $|R| + 3 + |T| + 1$ transitions. Thus, the size of M' is the order of the size of G. Altogether, the reduction can be done in deterministic logarithmic space.

The proof of (ii) is similar. First, the problem belongs again to P: We construct an NFA N for $P(M)$ and then have to test whether $P(M) \subseteq \Gamma^{\leq k}$. This question is equivalent to test whether $P(M) \cap \Gamma^{>k}$ is empty. For the set $\Gamma^{>k}$ we can construct a DFA with $k + 2$ states. Thus, an NFA N' accepting the intersection is of size $O(k|Q|^2|\Gamma|^2)$. Finally, we test the emptiness of N' which is known to be decidable in NLOGSPACE [17]. Altogether, the problem can be decided in polynomial time.

The P-completeness of the problem can be shown analogously as in (i). For a given context-free grammar G we construct again a PDA M' and observe that $P(M')$ is a finite set of words having at most length k if and only if $L(G)$ is empty. □

A PDA M is of *constant height* whenever there exists a constant $k \geq 1$ such that, for any word in $L(M)$, there exists an accepting computation along which the pushdown store never contains more than k symbols [2, 5]. As a consequence of Theorem 15, we get the P-completeness of testing constant height property for *unambiguous* PDA, i.e., presenting at most one accepting computation on any input:

Corollary 16. *Given an unambiguous PDA M, it is* P*-complete to decide whether: (i) M is a constant height PDA. (ii) M is a PDA of constant height k, for a given $k \geq 1$.*

The next P-completeness result can be shown similarly as Theorem 15.

Theorem 17. *Given a PDA M, it is* P*-complete to decide whether $P(M)$ is a subset of Z^*Z_0.*

Proof. To show that the problem belongs to P, we construct again an NFA for $P(M)$. Then we have to test whether there is exactly one $Z \in \Gamma$ such that $P(M) \subseteq Z^*Z_0$. For every $Z \in \Gamma$ such a test can be performed in NLOGSPACE

[17]. Altogether, we have to perform at most $|\Gamma|$ such tests. Thus, we obtain that deterministic polynomial time is sufficient to decide the problem.

To show the P-completeness, we use a similar approach as in Theorem 15. Let $G = \langle N, T, S, R\rangle$ be a context-free grammar. Let $\$, \& \notin T$ be new symbols and $S', S'' \notin N$ be new nonterminals. Then, we define another context-free grammar

$$G' = \langle N \cup \{S', S''\}, T \cup \{\$, \&\}, S', R \cup \{S' \to SS'', S'' \to S''\$ \mid S''\& \mid \$\}\rangle,$$

and observe that $L(G') = L(G)\$\{\$, \&\}^*$. From G' we construct again an equivalent PDA M' using the standard construction and, similar to the above considerations, we obtain that M' is a counter machine if and only if $L(G)$ is empty. Furthermore, the reduction can be done in deterministic logarithmic space. □

As a consequence, we get the P-completeness of deciding whether a PDA is *essentially* a counter machine, i.e., in all of its accepting computations the pushdown storage is used as a counter:

Corollary 18. *Given a PDA M, it is* P*-complete to decide whether M is essentially a counter machine.*

References

1. Autebert, J.-M., Berstel, J., Boasson, L.: Context-free languages and pushdown automata. In: Handbook of Formal Languages, Vol. 1. pp. 111–174. Springer (1997)
2. Bednárová, Z., Geffert, V., Mereghetti, C., Palano, B.: The size-cost of Boolean operations on constant height deterministic pushdown automata. In: DCFS 2011. LNCS 6808, pp. 80–92. Springer (2011)
3. Dassow, J., Manea, F., Truthe, B.: On contextual grammars with subregular selection languages. In: DCFS 2011. LNCS 6808, pp. 25–27. Springer (2011)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., San Francisco (1979)
5. Geffert, V., Mereghetti, C., Palano, B.: More concise representation of regular languages by automata and regular expressions. Inf. Comput. 208, 385–394 (2010)
6. Ginsburg, S., Greibach, S.A., Harrison, M.A.: Stack automata and compiling. J. ACM 14, 172–201 (1967)
7. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. UCS 8, 193–234 (2002)
8. Greibach, S.A.: A note on pushdown store automata and regular systems. Proc. Amer. Math. Soc. 18, 263–268 (1967)
9. Greibach, S.A.: An infinite hierarchy of context-free languages. J. ACM 16, 91–106 (1969)
10. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley, Reading, Massachusetts (1978)
11. Hartmanis, J.: Context-free languages and Turing machines computations. Proc. Symposium on Applied Mathematics 19, 42–51 (1967)
12. Hoogeboom, H.J., Engelfriet, J.: Pushdown automata. In: Formal Languages and Applications. Stud. Fuzziness Soft Comput., vol. 148. pp. 117–138. Springer (2004)
13. Holzer, M., Kutrib, M.: Nondeterministic finite automata—recent results on the descriptional and computational complexity. Int. J. Found. Comp. Sci. 20, 563–580 (2009)
14. Holzer, M., Kutrib, M.: Descriptional complexity—an introductory survey. In: Scientific Applications of Language Methods. pp. 1–58. Imperial College Press (2010)
15. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts (1979)

16. Immerman, N.: Nondeterministic space is closed under complementation. SIAM J. Comput. 17, 935–38 (1988)
17. Jones, N.D.: Space-bounded reducibility among combinatorial questions. J. Comput. System. Sci. 11, 68–85 (1975)
18. Jones, N.D., Laaser, W.T.: Complete problems for deterministic polynomial time.
19. Mereghetti, C., Palano, B.: Quantum finite automata with control language. Theoretical Informatics and Applications 40, 315–332 (2006)
20. Szelepcsényi, R.: The method of forced enumeration for nondeterministic automata. Acta Inform. 26, 279–84 (1988)
21. Yu, S.: Regular languages. In: Handbook of Formal Languages, Vol. 1. pp. 41–110. Springer (1997)

Institut für Informatik

Justus-Liebig-Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany

Recent Reports

(Further reports are available at `www.informatik.uni-giessen.de`.)

M. Holzer, S. Jakobi, *On the Complexity of Rolling Block and Alice Mazes*, Report 1202, March 2012.

M. Holzer, S. Jakobi, *Grid Graphs with Diagonal Edges and the Complexity of Xmas Mazes*, Report 1201, January 2012.

H. Gruber, S. Gulan, *Simplifying Regular Expressions: A Quantitative Perspective*, Report 0904, August 2009.

M. Kutrib, A. Malcher, *Cellular Automata with Sparse Communication*, Report 0903, May 2009.

M. Holzer, A. Maletti, *An $n \log n$ Algorithm for Hyper-Minimizing States in a (Minimized) Deterministic Automaton*, Report 0902, April 2009.

H. Gruber, M. Holzer, *Tight Bounds on the Descriptional Complexity of Regular Expressions*, Report 0901, February 2009.

M. Holzer, M. Kutrib, and A. Malcher (Eds.), *18. Theorietag Automaten und Formale Sprachen*, Report 0801, September 2008.

M. Holzer, M. Kutrib, *Flip-Pushdown Automata: Nondeterminism is Better than Determinism*, Report 0301, February 2003

M. Holzer, M. Kutrib, *Flip-Pushdown Automata: $k+1$ Pushdown Reversals are Better Than k*, Report 0206, November 2002

M. Holzer, M. Kutrib, *Nondeterministic Descriptional Complexity of Regular Languages*, Report 0205, September 2002

H. Bordihn, M. Holzer, M. Kutrib, *Economy of Description for Basic Constructions on Rational Transductions*, Report 0204, July 2002

M. Kutrib, J.-T. Löwe, *String Transformation for n-dimensional Image Compression*, Report 0203, May 2002

A. Klein, M. Kutrib, *Grammars with Scattered Nonterminals*, Report 0202, February 2002

A. Klein, M. Kutrib, *Self-Assembling Finite Automata*, Report 0201, January 2002

M. Holzer, M. Kutrib, *Unary Language Operations and its Nondeterministic State Complexity*, Report 0107, November 2001

A. Klein, M. Kutrib, *Fast One-Way Cellular Automata*, Report 0106, September 2001

M. Holzer, M. Kutrib, *Improving Raster Image Run-Length Encoding Using Data Order*, Report 0105, July 2001

M. Kutrib, *Refining Nondeterminism Below Linear-Time*, Report 0104, June 2001

M. Holzer, M. Kutrib, *State Complexity of Basic Operations on Nondeterministic Finite Automata*, Report 0103, April 2001

M. Kutrib, J.-T. Löwe, *Massively Parallel Fault Tolerant Computations on Syntactical Patterns*, Report 0102, March 2001

A. Klein, M. Kutrib, *A Time Hierarchy for Bounded One-Way Cellular Automata*, Report 0101, January 2001

M. Kutrib, *Below Linear-Time: Dimensions versus Time*, Report 0005, November 2000

M. Kutrib, *Efficient Universal Pushdown Cellular Automata and their Application to Complexity*, Report 0004, August 2000

M. Kutrib, J.-T. Löwe, *Massively Parallel Pattern Recognition with Link Failures*, Report 0003, June 2000

A. Klein, M. Kutrib, *Deterministic Turing Machines in the Range between Real-Time and Linear-Time*, Report 0002, February 2000