Fundamenta Informaticae XXI (2001) 1001–1020 1001
IOS Press

Blackhole Pushdown Automata

Erzsébet Csuhaj-Varjú*
Computer and Automation Research Institute, Hungarian Academy of Sciences
Kende u. 13–17, 1111 Budapest, Hungary
csuhaj@sztaki.hu

Tomáš Masopust†
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Institute of Mathematics, Czech Academy of Sciences, Žižkova 22, 61662 Brno, Czech Republic
masopust@ipm.cz

György Vaszil
Computer and Automation Research Institute, Hungarian Academy of Sciences
Kende u. 13–17, 1111 Budapest, Hungary
vaszil@sztaki.hu

Abstract. We introduce and investigate blackhole pushdown automata, variants of pushdown automata, where a string can always be pushed to the pushdown, but only a given depth of the pushdown content is remembered (the rest of the pushdown content is either canceled or becomes inaccessible). We also study blackhole variants of regulated pushdown automata, where the automaton in some distinguished states checks the form of its pushdown content against a given control language. We present characterizations of several language families in terms of these constructs.

Keywords: Pushdown automaton, regulation, computational power.

Address for correspondence: E. Csuhaj-Varjú, P.O. Box 63, 1518 Budapest, Hungary
*Also affiliated with the Department of Algorithms and Their Applications, Faculty of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/c, 1117 Budapest, Hungary
†Research supported by the GAČR grant no. P202/11/P028, and by the CAS, Inst. Research Plan no. AV0Z10190503.

1. Introduction

The concept of a pushdown automaton is one of the basic notions of automata theory which has been investigated for a long time. Recently, its regulated versions have obtained increased interest. One example is a variant discussed in [3]. In this model the automaton is given a control language over the alphabet of pushdown symbols, and an input string is accepted whenever the pushdown automaton accepts it by a computation where the pushdown content of each step forms a string included in the given control language. It was shown that if the control language is regular, then the computational power is the same as that of ordinary pushdown automata. On the other hand, an example was presented demonstrating that non-regular linear control languages increase the computational power of these automata; the question concerning their computational power with non-regular linear control languages was examined in [5].

Studying nondeterminism in pushdown automata, the previous modification of these constructs has been generalized, and the notion of a so-called R-PDA was introduced in [4]. Given a control language R, an R-PDA is a pushdown automaton which makes a nondeterministic step whenever the pushdown content is in R, and makes a deterministic step whenever the pushdown content is not in R. In [4], it was shown that regular control languages do not change the computational power of these constructs, that is, they are context-free. The statement that having non-regular linear control languages, R-PDA are computationally complete computation devices was proved in [5].

While R-PDA check the form of their pushdown content in each computational step, the so-called state-controlled R-PDA (R-sPDA), introduced and studied in [5], perform such a check only in some distinguished, so-called checking states. As for R-PDA, if R is a regular language, the computational power is that of PDA, while if R is non-regular linear, then these computational devices are as powerful as Turing machines. In addition, two checks of the form of the pushdown content are sufficient to accept any recursively enumerable language.

Continuing the previous research, in [1] the concept of a blackhole state-controlled R-PDA (blackhole R-sPDA, for short) was introduced, where a symbol can always be pushed to the pushdown, but only a given depth of the pushdown content is considered, the rest of the pushdown content is deleted. Furthermore, in checking states, the automaton should check whether or not its pushdown content is an element of the given control language R. The notion was motivated by an observation on the functioning of regulated pushdown automata, namely, that in many cases, some parts of the pushdown content are needed or useful only during a limited period of the computation, but never later. Furthermore, the question of the amount of required memory is the basic question in complexity theory.

In [1] the characterization of several language families accepted by blackhole R-sPDA was presented. It was shown that any context-sensitive language can be accepted by a blackhole R-sPDA where the control language is linear and the depth of the pushdown taken into consideration is a linear function of the length of the input word. It was also proved that every context-free language can be accepted by a blackhole R-sPDA with a regular control language and a linear depth function. However, there exists a non-context-free language that can be accepted by a blackhole R-sPDA with a regular control language and a sublogarithmic depth function.

In this paper, which is an extended revised version of [1], we make further steps. First, we introduce the blackhole variant of the ordinary pushdown automata, called blackhole pushdown automata where at any step of the computation only a given depth of the pushdown content is considered. This means that the blackhole pushdown automaton can always push a string to its pushdown, but only the topmost $f(n)$ symbols, for some function f, (where n is the length of the input) are kept in their original form. We

define two working modes for these constructs: in the strong mode - analogously to the model in [1] - the part of the pushdown content which is not considered anymore is deleted, while in the weak mode, this useless string is replaced by a special symbol, $\#$, which can never be read from the pushdown. Thus, the difference between the work of blackhole pushdown automata working in the weak or in the strong mode consists in the way they use the value $f(n)$ of the depth function which is given to them as an "oracle" from outside of the system. Using the weak working mode, a blackhole PDA can be considered as a usual pushdown automaton which only operates on the topmost $f(n)$ positions of the pushdown: if at least one symbol is pushed to a position deeper than $f(n)$, then the bottom of the pushdown, Z_0, cannot be reached anymore due to the appearance of symbol $\#$ and no transition with Z_0 as the topmost symbol in the pushdown can be performed during the following computational steps. Thus, in the weak mode, the pushdown "overflow" is recorded, but as Z_0 can never be reached, the pushdown store cannot be emptied anymore, or in other words, its emptiness cannot be checked anymore. In the strong mode, the length of the pushdown content is kept under a bound during the whole computation without recording the (possible) overflow, but with the possibility of still emptying the pushdown until the bottom symbol Z_0 is reached. This opens several possibilities in the strong mode: For example, to check whether the length of a sequence of letters is equal to $f(n) - 1$, the automaton might place two marker symbols to the bottom, then push the sequence into the pushdown, and finally check if the second marker is the last symbol preceding Z_0 in the pushdown.

Continuing the research started in [1], we also study the weak working mode for R-sPDA as well; the strong mode is the functioning mode used in that paper.

In this paper, we provide characterizations of language classes accepted by blackhole PDA and R-sPDA in the strong and weak modes. We show that there exist non-context-free languages that can be accepted by some blackhole PDA with a sublogarithmic depth function working in the strong mode and in the weak mode as well. We also present hierarchies of language classes defined by blackhole PDA which are based on a sublogarithmic hierarchy of depth functions. We prove that there exist context-free languages which cannot be accepted by any blackhole PDA with a sublinear depth function and any blackhole R-sPDA with a sublinear depth function and an arbitrary control language in either the strong or the weak working modes. There exist, however, non-context-free languages which can be accepted by blackhole PDA with a linear depth function and blackhole R-sPDA with a linear depth function and a regular control language. We also study the power of blackhole R-sPDA with linear control languages. We show that both in the strong and in the weak working modes these automata describe the family of context-sensitive languages if the depth function is linear. We also discuss the case of deterministic blackhole R-sPDA (where the underlying PDA is deterministic). Finally, we close the paper by conclusions and open problems for future research.

2. Preliminaries

Throughout the paper, we assume the reader to be familiar with the basics of automata and formal language theory; for further details we refer to [6, 7].

For a set A, $|A|$ designates the cardinality of A. For an alphabet (a finite nonempty set) V, V^* denotes the set of all words over V; the empty word is denoted by λ. We set $V^+ = V^* - \{\lambda\}$. For a string $w \in V^*$, the length of w is denoted by $|w|$, and the mirror image of w by w^R. For a language

$L \subseteq V^*$, the mirror image of L is the language $L^R = \{w^R : w \in L\}$. The set of all natural numbers (including zero) and the set of positive integers are denoted by $\mathbb{N}$ and $\mathbb{N}^+$, respectively.

A *pushdown automaton* (a PDA, for short) is a septuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where Q is a finite set of states, Σ is the input alphabet, Γ is the pushdown alphabet, $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Q \times \Gamma^*}$ is the transition function, $q_0 \in Q$ 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 $\mathcal{M}$ is a triplet (q, w, γ), where $q \in Q$ is the current state, $w \in \Sigma^*$ is the unread part of the input, and $\gamma \in \Gamma^*$ is the current content of the pushdown (the leftmost symbol of γ is the top pushdown symbol). For $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $\gamma, \beta \in \Gamma^*$, $Z \in \Gamma$, and $(p, \beta) \in \delta(q, a, Z)$, we say that $\mathcal{M}$ makes a move from $(q, aw, Z\gamma)$ to $(p, w, \beta\gamma)$, written as $(q, aw, Z\gamma) \vdash_{\mathcal{M}} (p, w, \beta\gamma)$. For the sake of simplicity, the initial pushdown symbol appears only at the bottom of the pushdown during any computation, i.e., if $(p, \beta) \in \delta(q, a, Z)$, then either $Z \neq Z_0$ and β does not contain Z_0, or $Z = Z_0$ and $\beta = \beta' Z_0$, where β' does not contain Z_0. As usual, the reflexive and transitive closure of the relation $\vdash_{\mathcal{M}}$ is denoted by $\vdash^*_{\mathcal{M}}$ (if no confusion arises, the subscript $\mathcal{M}$ may be omitted). The *language accepted* by $\mathcal{M}$ is defined by $L(\mathcal{M}) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^*_{\mathcal{M}} (q, \lambda, \gamma) \text{ for some } q \in F \text{ and } \gamma \in \Gamma^*\}$.

A pushdown automaton $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is *deterministic* (a DPDA, for short) if the following two conditions are satisfied: (1) $|\delta(q, a, Z)| \leq 1$, for all $a \in \Sigma \cup \{\lambda\}$, $q \in Q$, and $Z \in \Gamma$, and (2) for all $q \in Q$ and $Z \in \Gamma$, if $\delta(q, \lambda, Z) \neq \emptyset$, then $\delta(q, a, Z) = \emptyset$, for all $a \in \Sigma$. That is, $\mathcal{M}$ is able to make at most one move from any configuration. In this case, we write $\delta(q, a, Z) = (p, \gamma)$ instead of $\delta(q, a, Z) = \{(p, \gamma)\}$.

A *context-free grammar* is a quadruple $G = (N, T, P, S)$, where N is the alphabet of nonterminals, T is the alphabet of terminals, $N \cap T = \emptyset$, $S \in N$ is the start symbol, and P is a finite set of productions of the form $u \to v$, where $u \in N$ and $v \in (N \cup T)^*$. For two strings $x, y \in V^*$ and a production $u \to v \in P$, define the relation $xuy \Rightarrow xvy$. The language generated by G (the language of G, for short) is defined by $L(G) = \{w \in T^* : S \Rightarrow^* w\}$, where $\Rightarrow^*$ is the reflexive and transitive closure of the relation $\Rightarrow$. A language L is *context-free* if there exists a context-free grammar G such that $L = L(G)$.

A language is context-free if and only if it can be accepted by a pushdown automaton. A context-free language is called *deterministic* if it can be accepted by a deterministic pushdown automaton.

A context-free grammar G is said to be in *Greibach normal form* if all of its productions are of the form $u \to bv$, where $u \in N$, $b \in T$, and $v \in N^*$. If $\lambda \in L(G)$, then we also consider the rule $S \to \lambda$.

A *linear grammar* is a context-free grammar $G = (N, T, P, S)$, where all productions are of the form $u \to v$, where $u \in N$ and $v \in T^* \cup T^* N T^*$, i.e., v is a string containing not more than one nonterminal. A language L is *linear* if there exists a linear grammar G such that $L = L(G)$.

A *regular grammar* is a context-free grammar $G = (N, T, P, S)$, where all productions are of one of the forms $A \to aB$, $A \to a$, $A \to \lambda$, where $A, B \in N$ and $a \in T$. A language L is *regular* if there exists a regular grammar G such that $L = L(G)$.

A *linear bounded automaton* (an LBA) is a construct $\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, \rhd, \lhd)$ where Q is a finite set of states, $\Sigma \subseteq \Gamma$ is the input alphabet, Γ is the tape alphabet, $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$ is the transition function, $q_0 \in Q$ is the initial state, q_{acc} is the accepting state, and $\rhd, \lhd \in \Gamma$ are the left and right end markers, respectively. The transition function δ is defined in such a way that it never moves the head to the left or to the right of the left and right end markers, respectively, neither replaces them with another symbol.

A *configuration* of $\mathcal{K}$ is a string of the form $w_1 q w_2$, where $w_1 w_2$ represents the current content of the tape, $w_1 w_2 \in \{\rhd\}(\Gamma - \{\lhd, \rhd\})^*\{\lhd\}$, the reading head scans the leftmost symbol of w_2 (or $\lhd$ if

$w_2 = \lambda$), and $q \in Q$ is the current state of the machine. Given an input string w_0, the initial configuration is $q_0 \triangleright w_0 \triangleleft$.

For technical reasons, δ can also be given in the following form: if $(q, b, L) \in \delta(p, a)$, then we write $xpa \to qxb$, for all $x \in \Gamma$, and if $(q, b, R) \in \delta(p, a)$, then we write $pa \to bq$, i.e., the symbol read by the head is the symbol to the right of the state. For a configuration $w_1 xpaw_2$ and a rule $xpa \to qxb$, we may write the computational step relation $\vdash$ so that $w_1 xpaw_2 \vdash w_1 qxbw_2$. Analogously, for $pa \to qp$, we write $w_1 xpaw_2 \vdash w_1 xbqw_2$.

An *accepting* computation on the input string w_0 is a sequence of computational steps $u_0 \vdash u_1 \vdash \cdots \vdash u_n$, for some $n \geq 1$, such that u_0 is the initial configuration $q_0 \triangleright w_0 \triangleleft$, u_i are configurations for all $i = 1, 2, \ldots, n-1$, and u_n is a configuration of the form $w_1 q_{acc} w_2$. An input string w_0 is *accepted* if there is an accepting computation on w_0.

A language is context-sensitive if and only if it is accepted by an LBA.

The classes of regular, linear, (deterministic) context-free, context-sensitive and recursively enumerable languages are denoted by REG, LIN, (D)CF, CS, and RE, respectively.

3. Basic Definitions and Properties

We first introduce the notion of a blackhole pushdown automaton and define its two functioning modes, the weak and the strong working modes. Then, we recall the notion of a blackhole R-sPDA from [1], and describe its strong and weak working modes. We note that the strong mode corresponds to the functioning mode introduced in [1]. We close the section with some basic results concerning language classes accepted by blackhole PDA and blackhole R-sPDA.

Definition 3.1. A pair $\mathcal{M} = (\mathcal{M}', f)$ is called a *blackhole pushdown automaton* (a blackhole PDA, for short), if $\mathcal{M}' = (Q, \Sigma, \bar{\Gamma}, \delta, q_0, Z_0, F)$ is a pushdown automaton such that $\bar{\Gamma} = \Gamma \cup \{\#\}$ where $\# \notin \Gamma$ and $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Q \times \Gamma^*}$. The function $f : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ is called the *depth function*.

Notice that δ is defined in such a way that the symbol $\#$ cannot be read from the pushdown.

Before describing how a blackhole pushdown automaton $\mathcal{M}$ functions, we need two auxiliary notions. We define the function $\text{trim}_n : \Gamma^* \to \Gamma^*$ for $n \in \mathbb{N} \cup \{\infty\}$ by

$$\text{trim}_n(\alpha Z_0) = \begin{cases} \alpha Z_0 & \text{if } |\alpha| \leq n, \text{ or } n = \infty, \\ \alpha_1 Z_0 & \text{if } \alpha = \alpha_1 \alpha_2 \text{ with } |\alpha_1| = n, \alpha_2 \in \Gamma^*. \end{cases}$$

Let the function $\text{top}_n : \bar{\Gamma}^* \to \bar{\Gamma}^*$ for $n \in \mathbb{N} \cup \{\infty\}$ be defined as follows:

$$\text{top}_n(\alpha Z_0) = \begin{cases} \alpha Z_0 & \text{if } \alpha \in \Gamma^* \cup \Gamma^*\{\#\}, |\alpha| \leq n, \text{ or } n = \infty \\ \alpha_1 \# Z_0 & \text{if } \alpha = \alpha_1 \alpha_2, \ \alpha_1 \in \Gamma^*, \ |\alpha_1| = n, \ \alpha_2 \in \Gamma^+ \cup \Gamma^*\{\#\}. \end{cases}$$

A blackhole pushdown automaton $\mathcal{M}$ has two functioning modes, namely, the *strong* mode (or the s-mode, for short) and the *weak* mode (or the w-mode, for short). Let $\mathcal{M}$ be a blackhole PDA, $w_0 \in \Sigma^*$ be an input word, $q, q' \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \bar{\Gamma}^*$.

A move performed by $\mathcal{M}$ in the *strong mode* on configuration $(q, aw, Z\gamma)$ is defined as follows:

$$(q, aw, Z\gamma) \vdash^s_{\mathcal{M}} (q', w, \text{trim}_{f(|w_0|)}(\gamma'\gamma))$$

where $(q', \gamma') \in \delta(q, a, Z)$.

A move performed by $\mathcal{M}$ in the *weak mode* on the configuration $(q, aw, Z\gamma)$ is defined as follows:

$$(q, aw, Z\gamma) \vdash^{w}_{\mathcal{M}} (q', w, \text{top}_{f(|w_0|)}(\gamma'\gamma))$$

where $(q', \gamma') \in \delta(q, a, Z)$.

The reflexive and transitive closure of $\vdash^{X}_{\mathcal{M}}$, $X \in \{w, s\}$, is denoted by $\vdash^{X^*}_{\mathcal{M}}$.

Blackhole pushdown automata can be equipped with further regulations. For example, they can be given with distinguished checking and non-checking states and a control language.

A triplet $\mathcal{M} = (\mathcal{M}', Q_c, R)$ is called a *blackhole state-controlled R-PDA* (a blackhole R-sPDA, for short), if $\mathcal{M}' = (\mathcal{M}'', f)$ is a blackhole pushdown automaton, $Q_c \subseteq Q$ is a subset of states of $\mathcal{M}''$ called the set of *checking states* of $\mathcal{M}$, and $R \subseteq (\Gamma - \{Z_0\})^*$ is a *control language*, where Γ is the set of pushdown symbols of $\mathcal{M}''$.

Blackhole state-controlled R-PDA automata work analogously to blackhole pushdown automata, except that in the checking states the content of their pushdown up to the given depth is checked according to their membership in the given control language R. In the strong mode, the whole pushdown content, in the weak mode only a prefix of the pushdown content is checked for membership in R. This is in accordance with the previous definitions: blackhole PDA cannot reach the bottom of the pushdown in the weak mode, while blackhole R-sPDA cannot "see" the bottom of the pushdown in the weak mode.

Let $\mathcal{M}$ be a blackhole R-sPDA, $w_0 \in \Sigma^*$ be an input word, $q, q' \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$.

A move performed by $\mathcal{M}$ on its configuration $(q, aw, Z\gamma)$ in the *strong mode* (in the s-mode, for short) is defined as

$$(q, aw, Z\gamma) \vdash^{s}_{\mathcal{M}} (q', w, \text{trim}_{f(|w_0|)}(\gamma'\gamma))$$

where $(q', \gamma') \in \delta(q, a, Z)$, and either $q \in Q - Q_c$, or $q \in Q_c$ and $Z\gamma = \gamma'' Z_0$ with $\gamma'' \in R$.

A move performed by $\mathcal{M}$ on its configuration $(q, aw, Z\gamma)$ in the *weak mode* (in the w-mode, for short) is defined as

$$(q, aw, Z\gamma) \vdash^{w}_{\mathcal{M}} (q', w, \text{top}_{f(|w_0|)}(\gamma'\gamma))$$

where $(q', \gamma') \in \delta(q, a, Z)$, and either $q \in Q - Q_c$, or $q \in Q_c$ and $Z\gamma = \gamma''\gamma''' Z_0$ with $\gamma'' \in R$, and $\gamma''' \in \Gamma^* \cup \Gamma^*\{\#\}$.

Notice that during the computation, Z_0 always remains in the pushdown both in the strong and in the weak working modes and in the case of both blackhole PDA and blackhole R-sPDA.

Informally, a blackhole PDA (a blackhole state-controlled R-PDA) works in such a way that it can always push a string to its pushdown (it may be the empty string), but only the topmost $f(n)$ symbols (where n is the length of the input) are kept in the pushdown in their original form. The word at the bottom of the pushdown consisting of the symbols which, at least for one computational step during the computation occupy a position greater than $f(n)$, are either denoted by the single symbol $\#$ (when the automaton works in the weak mode), or they are deleted (when the automaton works in the strong mode). This is realized by the use of the top function which changes all the symbols which have occupied a pushdown position greater than $f(n)$ to a single $\#$ (in the case of weak mode), and by the use of the trim function which deletes the letters which occupy position greater than $f(n)$ (in the case of strong mode).

As we mentioned in the Introduction, the difference between the work of blackhole pushdown automata working in the weak or in the strong mode, consists in the way they use the value $f(n)$ of the

depth function. Using the weak working mode, a blackhole PDA can be considered as a usual pushdown automaton which only operates on the topmost $f(n)$ positions of the pushdown: if at least one symbol is pushed to a position deeper than $f(n)$, then the bottom of the pushdown, Z_0, cannot be reached anymore, due to the appearance of symbol $\#$, thus, the automaton can never make sure that its pushdown is emptied. Working in the strong mode, however, the automaton can check, for example, by emptying its pushdown store, if a symbol pushed to the pushdown previously is still present or already erased by the application of the blackhole principle. This also holds for the state controlled variants: in the strong mode the whole pushdown content, in the weak mode only a prefix of the pushdown content is checked for membership in the control language.

If $\mathcal{M}''$ is a deterministic pushdown automaton, then we call any blackhole PDA $\mathcal{M}' = (\mathcal{M}'', f)$ and any blackhole R-sPDA $\mathcal{M} = (\mathcal{M}', Q_c, R)$ *deterministic*.

Analogously to ordinary pushdown automata, the *language* accepted by a blackhole pushdown automaton (a blackhole state-controlled R-PDA) $\mathcal{M}$ in the X-mode, where $X \in \{w, s\}$, is defined by

$$L_X(\mathcal{M}) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash_{\mathcal{M}}^{X^*} (q, \lambda, \gamma) \text{ for some } q \in F \text{ and } \gamma \in (\Gamma \cup \{\#\})^*\}.$$

Let $\mathcal{L}_X(\text{PDA}, \mathcal{F})$ and $\mathcal{L}_X(\text{PDA}, \mathcal{F}, \mathcal{C})$, for $X \in \{s, w\}$, $\mathcal{F}$ being a class of functions with domain $\mathbb{N} \cup \{\infty\}$, and $\mathcal{C}$ being a class of control languages, denote the language classes which are accepted by blackhole PDA or blackhole R-sPDA working in the strong or in the weak mode (for $X = s$ or $X = w$, respectively) with depth function from the class $\mathcal{F}$ (or unbounded depth, in case of $\mathcal{F} = \infty$) and with control language R from the language class $\mathcal{C}$ in case of blackhole R-sPDA. We replace PDA in the above notation by DPDA if the blackhole PDA (or blackhole R-sPDA) is deterministic.

There are several properties of blackhole pushdown automata which can easily be seen by the definitions.

Proposition 3.1. $\mathcal{L}_X(\text{PDA}, \mathcal{F}) \subseteq \mathcal{L}_X(\text{PDA}, \mathcal{F}, \mathcal{C})$ for $X \in \{w, s\}$, any class of depth functions $\mathcal{F}$ and any nonempty class of control languages $\mathcal{C}$.

To see this, consider for each blackhole PDA $\mathcal{M}$, the blackhole R-sPDA $\mathcal{M}' = (\mathcal{M}, \emptyset, R)$ where R is an arbitrary language from $\mathcal{C}$.

We also immediately obtain the following statement by the definitions.

Proposition 3.2. $\mathcal{L}_s(\text{PDA}, \infty, \mathcal{C}) = \mathcal{L}_w(\text{PDA}, \infty, \mathcal{C})$ holds for any class of control languages $\mathcal{C}$.

Now we show that blackhole PDA in the strong mode are at least as powerful as blackhole PDA working in the weak mode.

Lemma 3.1. The following statements hold for arbitrary depth functions $f : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$.

1. For any blackhole PDA $\mathcal{M}_1 = (\mathcal{M}_1', f)$, there is a blackhole PDA $\mathcal{M}_2 = (\mathcal{M}_2', f)$ such that $L_w(\mathcal{M}_1) = L_s(\mathcal{M}_2)$.

2. For any blackhole R-sPDA $\mathcal{M}_1 = (\mathcal{M}_1', Q_c, R)$ with $\mathcal{M}_1' = (\mathcal{M}_1'', f)$, there exists a blackhole R-sPDA $\mathcal{M}_2 = (\mathcal{M}_2', Q_c, R')$ with $\mathcal{M}_2' = (\mathcal{M}_2'', f)$ such that $L_w(\mathcal{M}_1) = L_s(\mathcal{M}_2)$ and $R' = R \cdot \Gamma^*$, where Γ is the pushdown alphabet of $\mathcal{M}_2$.

Proof:
Consider the blackhole PDA $\mathcal{M}_1 = (\mathcal{M}'_1, f)$, $\mathcal{M}'_1 = (Q_1, \Sigma, \Gamma_1 \cup \{\#\}, \delta_1, q_0, Z_0, F)$, which accepts in the weak mode the language L, i.e., $L = L_w(\mathcal{M}_1)$. We construct a blackhole PDA $\mathcal{M}_2 = (\mathcal{M}'_2, f)$ such that $L = L_s(\mathcal{M}_2)$.

Let $\mathcal{M}'_2 = (Q_2, \Sigma, \Gamma_2 \cup \{\#\}, \delta_2, q'_0, Z_0, F)$ where $Q_2 = Q_1 \cup \{q'_0\}$, $\Gamma_2 = \Gamma_1 \cup \{[\lambda, Z'_0]\} \cup \{[Z, Z'_0] : Z \in \Gamma_1 - \{Z_0\}\}$, for a new symbol Z'_0, and δ_2 is defined as follows. First, let

$$\delta_2(q'_0, \lambda, Z_0) = (q_0, [\lambda, Z'_0]Z_0),$$

where q_0 is the initial state of $\mathcal{M}_1$. In this first move, $\mathcal{M}_2$ places the special symbol $[\lambda, Z'_0]$ to the bottom of the pushdown store, just above Z_0. This special symbol is used by $\mathcal{M}_2$ in the same way as Z_0 is used by $\mathcal{M}_1$, that is, for all $q \in Q_1$, $a \in \Sigma \cup \{\lambda\}$, and for all $(r, \gamma) \in \delta_1(q, a, Z_0)$, let

$$(r, \gamma') \in \delta_2(q, a, [\lambda, Z'_0]),$$

where $\gamma' = [\lambda, Z'_0]$ if $\gamma = Z_0$, and $\gamma' = \gamma''[Z, Z'_0]$ if $\gamma = \gamma'' Z Z_0$ for some γ'' and $Z \neq \lambda$. If a symbol Z is pushed on the top of Z_0 by $\mathcal{M}_1$, then these rules also replace $[\lambda, Z'_0]$ by $[Z, Z'_0]$, so the last two symbols in the pushdown of $\mathcal{M}_1$ are represented by a single symbol in $\mathcal{M}_2$.

For all $q \in Q_1$, $a \in \Sigma \cup \{\lambda\}$, $Z \in \Gamma_1 - \{Z_0\}$, and $(r, \gamma) \in \delta_1(q, a, Z)$, let

$$(r, \gamma') \in \delta_2(q, a, [Z, Z'_0])$$

with $\gamma' = \gamma''[Z', Z'_0]$ if $\gamma = \gamma'' Z'$, $Z' \in \Gamma_1$, or $\gamma' = [\lambda, Z_0]$ if $\gamma = \lambda$. These rules are necessary to make $\mathcal{M}_2$ work with $[Z, Z'_0]$ in the same way as $\mathcal{M}_1$ works with Z.

Finally, for all $q \in Q_1$, $a \in \Sigma \cup \{\lambda\}$, $Z \in \Gamma_1 - \{Z_0\}$, let

$$\delta_2(q, a, Z) = \delta_1(q, a, Z),$$

that is, those transition rules of $\mathcal{M}_2$ which do not use the special symbols are the same as the transitions of $\mathcal{M}_1$.

Consider a computation of the blackhole PDA $\mathcal{M}_1$ on an input word w. The machine $\mathcal{M}_2$ is constructed in such a way that the symbol $\#$ appears in the pushdown of $\mathcal{M}_1$ if and only if the symbol of the form $[Z, Z'_0]$ for some $Z \in \Gamma_1 - \{Z_0\}$ disappears from the bottom of the pushdown of $\mathcal{M}_2$ while reading the same input. This means that we have two cases. If Z_0 is reachable in the pushdown of $\mathcal{M}_1$ (there is no $\#$ above it), then $[Z, Z'_0]$ is present in the pushdown of $\mathcal{M}_2$, and all transitions of $\mathcal{M}_1$ reading Z_0 can be simulated by $\mathcal{M}_2$ reading $[\lambda, Z'_0]$. If Z_0 becomes unreachable during the computation of $\mathcal{M}_1$ (there is a $\#$ symbol appearing above it), then the symbol of the form $[Z, Z'_0]$ disappears from the pushdown of $\mathcal{M}_2$, and the rest of the content of the two pushdowns are the same. In $\mathcal{M}_1$, there is no transition for reading $\#$, in $\mathcal{M}_2$ there is no transition for reading Z_0, so the weak and strong computations of the two machines coincide.

Moreover, any check of the pushdown content of $\mathcal{M}_1$ in the weak mode for membership in the control language R gives the same result as the check by $\mathcal{M}_2$ in the strong mode for membership in the control language $R \cdot \Gamma_2^*$. □

4. Blackhole PDA and Blackhole State-Controlled R-PDA with Regular Control Languages

If the depth of the pushdown of a blackhole pushdown automaton $\mathcal{M}'$ is bounded by a constant, i.e., $f(n) = k$, for some $k \in \mathbb{N}$ and for all $n \in \mathbb{N}$, then both in the strong mode and in the weak mode the accepted language is regular. To see this, notice that a pushdown with depth bounded by a constant, say k, can only store a finite number of different strings, namely, $m = 1 + \sum_{j=1}^{k} |\Gamma|^j$ words for Γ being the pushdown alphabet. Therefore, if Q is the set of states of the underlying pushdown automaton $\mathcal{M}$, we can construct a finite automaton with the set of states $Q \times m$ which can keep track of the content of the pushdown, and thus simulate the functioning of $\mathcal{M}'$ in the strong mode. The same argumentation holds for the weak working mode as well.

For the case of blackhole R-sPDA, increasing the complexity of the control language R, we do not obtain larger accepting power, because R can only have a finite subset of the set of all strings of length at most k, so we can construct a finite automaton simulating the given blackhole R-sPDA working in the given mode. Thus, we have the following statement.

Proposition 4.1. $\mathcal{L}_X(\text{PDA}, O(1)) = \mathcal{L}_X(\text{PDA}, O(1), \mathcal{C}) = \text{REG}$ for $X \in \{w, s\}$ and any class of control languages $\mathcal{C}$.

Now we consider the case when we increase the available space, but remain below the logarithmic depth, that is, we have a depth function $f(n) \in o(\log n)$. According to [8, Theorem 5.2.1], the so-called strongly space-bounded one-way Turing machines accept only regular languages when sublogarithmic space is used (in the terminology of [8], a Turing machine is strongly $f(n)$ space bounded if there is no accessible configuration which uses more than $f(n)$ space). This theorem, however, is not applicable to the case of blackhole pushdown automata since the value of the depth function is known to these automata without using any of their computational resources. This observation is demonstrated by the following statement.

Theorem 4.1. There exists a language $L \in \mathcal{L}_w(\text{PDA}, o(\log n)) \cap \mathcal{L}_s(\text{PDA}, o(\log n))$ such that $L \notin \text{CF}$.

Proof:
Let us consider the non-context-free language $L = \{a^i b^j : j < \lfloor \log\log(i+j) \rfloor, i, j \geq 1\}$. Since $\mathcal{L}_w(\text{PDA}, o(\log n)) \subseteq \mathcal{L}_s(\text{PDA}, o(\log n))$, it is sufficient to show that L can be accepted by a blackhole PDA working in the weak mode. To this aim, let us construct $\mathcal{M} = (\mathcal{M}', f)$ with the sublogarithmic depth function $f(n) = \lfloor \log\log n \rfloor$.

Let $\mathcal{M}' = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{a, b, Z_0\} \cup \{\#\}, \delta, q_0, Z_0, \{q_3\})$ be a PDA with

$$\begin{aligned}
&\delta(q_0, a, Z) = (q_0, aZ),\ Z \in \{a, Z_0\}, && \delta(q_2, \lambda, b) = (q_2, \lambda),\\
&\delta(q_0, b, a) = (q_1, ba), && \delta(q_2, \lambda, a) = (q_3, a).\\
&\delta(q_1, b, b) = \{(q_1, bb), (q_2, bb)\}, &&
\end{aligned}$$

Processing a given input word w of length n, the automaton first reads and pushes the whole word into the pushdown. Notice that $\mathcal{M}$ can enter the final state q_3 if and only if $w \in \{a\}^+\{b\}^+$ and the number of b's is less than $\lfloor \log\log n \rfloor$. If this is the case, then at least one symbol a must be present in the pushdown. This is checked by popping the pushdown content and entering the final state q_3 after at least one a is read from the pushdown. □

Using similar ideas as those in the proof of Theorem 4.1, we can, in some sense, compute a whole class of functions using blackhole PDA working in the strong mode.

Theorem 4.2. Let $g : \mathbb{N}^+ \to \mathbb{N}^+$ be a function such that $0 < g(n) < n$, for all $n \in \mathbb{N}^+$. If $L = \{w \in \{a,b\}^+ : w = a^i b^j, i,j \geq 1, j = g(|w|)\}$, then $L = L_s(\mathcal{M}, f)$ for some blackhole PDA $\mathcal{M}$ with $f(n) = g(n) + 1$.

Proof:
Consider the blackhole PDA $\mathcal{M} = (\mathcal{M}', f)$ with depth function $f(n) = g(n) + 1$ where $\mathcal{M} = (Q, \{a,b\}, \{a,b,Z_0\} \cup \{\#\}, \delta, q_0, Z_0, \{q_4\})$ with $Q = \{q_0, q_1, q_2, q_3, q_4\}$, and

$$\begin{aligned}
\delta(q_0, a, Z) &= (q_0, aZ),\ Z \in \{a, Z_0\}, & \delta(q_2, \lambda, b) &= (q_2, \lambda),\\
\delta(q_0, b, a) &= (q_1, ba), & \delta(q_2, \lambda, a) &= (q_3, \lambda),\\
\delta(q_1, b, b) &= \{(q_1, bb), (q_2, bb)\}, & \delta(q_3, \lambda, Z_0) &= (q_4, Z_0).
\end{aligned}$$

First, note that if the input is not of the form $a^i b^j$, $i,j \geq 1$, then the automaton cannot reach state q_2, thus neither state q_4, which is the sole accepting state.

Assume that the input is of the form $w = a^i b^j$, where $i,j \geq 1$. After reading and pushing the whole word into the pushdown, the automaton is in state q_2, and there is a single a as the last symbol on the bottom above the initial pushdown symbol Z_0 if and only if the relationship of the length of the sequence of b's and the length of the whole word w is correct, that is, if and only if $j = g(|w|) = g(j+i)$. This is checked so that the automaton pops all symbols and enters the final state q_4 if only one a is present and there is no other symbol between the last b and Z_0. Since no other way of functioning is possible, $L_s(\mathcal{M}) = L$ holds. □

Now we show that there is a hierarchy of languages characterized by blackhole PDA which is based on a sublogarithmic hierarchy of depth functions. In the following we use the notation $\log^{(k)} n$, for some $k, n \geq 1$, to denote the composition of the logarithmic function, that is, $\log^{(k+1)} n = \log \log^{(k)} n$ where $\log^{(1)} n = \log n$.

Theorem 4.3. For any $k \geq 1$, there is a language L_k such that $L_k \in \mathcal{L}_w(\text{PDA}, O(\log^{(k)} n)) \cap \mathcal{L}_s(\text{PDA}, O(\log^{(k)} n))$, but $L_k \notin \mathcal{L}_w(\text{PDA}, O(\log^{(k+1)} n)) \cup \mathcal{L}_s(\text{PDA}, O(\log^{(k+1)} n))$.

Proof:
Consider the language $L_k = \{w = \$^{i-1} u \$ u^R : u \in \{a,b\}^+, \text{ and } |u| \leq \lfloor \log^{(k)}(|w|) \rfloor - 1\}, i \geq 2$, that is, the length of the subwords $\$u$ of the words of L_k are shorter or equal to the kth logarithm of the length of the whole words. First we construct, for each $k \geq 1$, a blackhole PDA $\mathcal{M}_k = (\mathcal{M}'_k, f)$ with the depth function $f(n) = \lfloor \log^{(k)} n \rfloor$ accepting L_k in the weak mode.

Since $\mathcal{L}_w(\text{PDA}, O(\log^{(k)} n)) \subseteq \mathcal{L}_s(\text{PDA}, O(\log^{(k)} n))$ (see Lemma 3.1), it is sufficient to show that $L_k \in \mathcal{L}_w(\text{PDA}, O(\log^{(k)} n)) \cap \mathcal{L}_s(\text{PDA}, O(\log^{(k)} n))$.

Let $\mathcal{M}'_k = (\{q_0, q_1, q_2, q_3\}, \{a,b,\$\}, \{a,b,\$,Z_0\} \cup \{\#\}, \delta, q_0, Z_0, \{q_3\})$ be a PDA with

$$\begin{aligned}
\delta(q_0, \$, Z) &= (q_0, \$Z),\ Z \in \{\$, Z_0\}, & \delta(q_1, x, Z) &= (q_1, xZ),\ x, Z \in \{a,b\},\\
\delta(q_0, x, \$) &= (q_1, x\$),\ x \in \{a,b\}, & \delta(q_2, x, x) &= (q_2, \lambda),\ x \in \{a,b\},\\
\delta(q_1, \$, Z) &= (q_2, Z),\ Z \in \{a,b\}, & \delta(q_2, \lambda, \$) &= (q_3, \$).
\end{aligned}$$

Processing an input word w of length n, $\mathcal{M}_k$ pushes the first group of \$'s and the subword $u \in \{a,b\}^+$ into the pushdown, and reads the last \$. The pushdown store contains the word $\bar{u}^R\$^l$, where $\bar{u}$ is a suffix of u and $l \geq 0$, followed possibly by a symbol $\#$, and then Z_0. If $|\$u| \leq \lfloor \log^{(k)}(|w|) \rfloor$ where $|w| = i + 2 \cdot |u|$, then there is at least one copy of \$ which is not deleted from the pushdown, i.e., $\bar{u}^R\$^l = u^R\$^m$, for $m \geq 1$. Then, the equality of the rest of the input with u^R is checked by reading and popping the pushdown in each step. If the read and popped symbols match and at least one \$ is found at the bottom of the pushdown, the automaton enters the final state q_3. On the other hand, if $|\$u| > \lfloor \log^{(k)}(|w|) \rfloor$, then the pushdown store contains no \$ because, due to the depth function, there is not enough space. Hence, the automaton never reaches state q_3.

To prove the rest of the statement, it is sufficient (by Lemma 3.1) to show that L_k cannot be accepted by a blackhole PDA working in the strong mode with depth function f in $O(\log^{(k+1)} n)$. Let us consider words $\$^{i-1}u\$u^R \in L_k$ where $|u| = \lfloor \log^{(k)}(|w|) \rfloor - 1$, $|w| = i + 2 \cdot |u|$. There are $2^{|u|} = 2^{\lfloor \log^{(k)}(|w|) \rfloor}$ different strings of this length, but the blackhole PDA has only $c \cdot 2^{\lfloor \log^{(k+1)}(|w|) \rfloor}$ different configurations, for some constant c. Therefore, if we take $\$^{i-1}u\$u^R \in L_k$, then there must be a word $\$^{i-1}v\$v^R \in L_k$ with $|u| = |v|$, but $u \neq v$, such that upon reading the last \$, the automaton is in the same configuration for both of these input words. This means that processing the word $\$^{i-1}u\$v^R \notin L_k$, the automaton also enters the final state, which is a contradiction. □

So far we have examined systems with sublogarithmic depth functions. Now we move on studying the cases when the depth function is linear.

Theorem 4.4. CF $\subset \mathcal{L}_X(\text{PDA}, O(n)) \subseteq \mathcal{L}_X(\text{PDA}, O(n), \text{REG}),\ X \in \{w, s\}$.

Proof:
It is clear by Proposition 3.1 that $\mathcal{L}_X(\text{PDA}, O(n)) \subseteq \mathcal{L}_X(\text{PDA}, O(n), \text{REG})$, for any $X \in \{w, s\}$. Moreover, CF $\subseteq \mathcal{L}_X(\text{PDA}, O(n))$, for $X \in \{w, s\}$, since for any context-free language L, there is a context-free grammar in Greibach normal form generating L. As every grammar in the Greibach normal form contains no erasing productions and the right-hand side of all of its rules starts with a terminal symbol, using the standard technique for the construction of a pushdown automaton based on context-free grammars, we obtain that there is a pushdown automaton $\mathcal{M}$ accepting L requiring not more than linear pushdown depth.

In addition, $\mathcal{L}_w(\text{PDA}, O(n)) \subseteq \mathcal{L}_s(\text{PDA}, O(n))$, that is, $\mathcal{L}_w(\text{PDA}, O(n)) \cap \mathcal{L}_s(\text{PDA}, O(n)) = \mathcal{L}_w(\text{PDA}, O(n))$ by Lemma 3.1. Thus, we only need to show that CF $\subset \mathcal{L}_w(\text{PDA}, O(n))$. To see this, let us consider the non-context-free language $L = \{a^i b^j c^j : i > j \geq 1\}$ and the depth function $f(n) = \lceil \frac{n}{3} \rceil$. Let $\mathcal{M} = (\mathcal{M}', f)$ where $\mathcal{M}' = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{a, b, c, Z_0\} \cup \{\#\}, \delta, q_0, Z_0, \{q_3\})$ with

$$\begin{aligned}
\delta(q_0, a, Z) &= (q_0, aZ),\ Z \in \{a, Z_0\}, & \delta(q_1, c, b) &= (q_2, \lambda),\\
\delta(q_0, b, a) &= (q_1, ba), & \delta(q_2, c, b) &= (q_2, \lambda),\\
\delta(q_1, b, b) &= (q_1, bb), & \delta(q_2, \lambda, a) &= (q_3, a).
\end{aligned}$$

Let $\mathcal{M}$ work in the weak mode. First, it reads and pushes a's and then reads and pushes b's to the pushdown. Then, in state q_2, it reads c's and pops b's and checks if the number of b's is equal to the number of c's in the input word. $\mathcal{M}$ accepts the input if, in addition, there is still at least one a in the pushdown, because this means that $i > j$, thus, $L = L_w(\mathcal{M})$ and the statement is proved. □

Recall, that there exist context-free languages which cannot be accepted by a one-way Turing machine in less than linear space, see [2, Theorem 11.4]. It is not obvious, however, that there also exist context-free languages which cannot be accepted by any sublinear depth blackhole PDA or R-sPDA, because these machines do not need to use their computational resources to obtain the values of the depth functions, which might increase their accepting power (as in the case of sublogarithmic depth blackhole PDA versus sublogarithmic space Turing machines, see Theorem 4.1).

Proposition 4.2. There exists a language $L \in$ CF such that $L \notin \mathcal{L}_X(\text{PDA}, o(n), \mathcal{C})$ for $X \in \{w, s\}$ and for any class of control languages $\mathcal{C}$.

Proof:
This can be seen similarly to the proof of [2, Theorem 11.4]. Consider the context-free language $L = \{wcw^R : w \in \{a, b\}^+\}$. If L is accepted by a blackhole R-sPDA $\mathcal{M}$ in any of the weak or strong modes, then after reading the prefix wc of a word wcw^R, $\mathcal{M}$ has to be able to reach 2^l different configurations, for $l = |w|$. However, if Q is the set of states and Γ is the pushdown alphabet, then the number of different configurations is $|Q| \cdot |\Gamma|^{f(2l+1)}$ which is less than 2^l for a sufficiently large l. □

Thus, linear pushdown depth is not only sufficient, but also necessary to accept all context-free languages.

5. Linear Depth Functions, Linear Control Languages

In this section we study the power of blackhole R-sPDA with linear control languages and linear depth functions.

If the depth of the pushdown is not bounded, i.e., $f(n) = \infty$, then blackhole R-sPDA coincide with state-controlled R-PDA which accept any recursively enumerable language with linear control languages and not more than two checks of the pushdown content [5]. Since, by definition, $\mathcal{L}_s(\text{PDA}, \infty, \text{LIN}) = \mathcal{L}_w(\text{PDA}, \infty, \text{LIN})$ holds, the following statement immediately follows.

Proposition 5.1. $\mathcal{L}_X(\text{PDA}, \infty, \text{LIN}) = \text{RE}$, for $X \in \{w, s\}$.

Now we consider blackhole R-sPDA with linear control languages and with linear depth functions. We show that both in the strong and in the weak working modes these automata describe the family of context-sensitive languages.

To illustrate our proof technique, we first present an example language which can be accepted by a blackhole R-sPDA working in the strong or in the weak mode.

Example 5.1. Let $L = \{ww : w \in \{a, b\}^+\}$. We construct a blackhole R-sPDA $\mathcal{M}$ with a linear control language R and with a linear depth function f. First we show that $\mathcal{M}$ accepts L working in the strong mode.

Let us define $\mathcal{M} = (\mathcal{M}', Q_c, R)$ such that $\mathcal{M}'$ is a blackhole PDA with $\mathcal{M}' = (\mathcal{M}'', f)$ and $\mathcal{M}'' = (\{q_0, q_1, q_c, q_2, q_3, q_f\}, \{a, b\}, \{a, b, Z_0, \$\} \cup \{\#\}, \delta, q_0, Z_0, \{q_f\})$ where $Q_c = \{q_c\}$ is the set of

checking states, $R = \{w\$w^R : w \in \{a,b\}^+\}$, and $f(n) = n+1$ is the (linear) depth function. The transition function δ is defined as follows. For $x, y \in \{a,b\}$,

$$\begin{array}{llll} \delta(q_0,x,Z_0) = (q_0,xZ_0), & \delta(q_1,\lambda,\$) = (q_1,x\$), & \delta(q_c,\lambda,x) = (q_2,x), & \delta(q_3,\lambda,x) = (q_3,\lambda), \\ \delta(q_0,x,y) = (q_0,xy), & \delta(q_1,\lambda,y) = (q_1,xy), & \delta(q_2,x,x) = (q_2,\lambda), & \delta(q_3,\lambda,Z_0) = (q_f,Z_0). \\ \delta(q_0,\lambda,y) = (q_1,\$y), & \delta(q_1,\lambda,x) = (q_c,x), & \delta(q_2,\lambda,\$) = (q_3,\lambda), & \end{array}$$

The automaton works as follows. Let us consider an input string w'. Then $\mathcal{M}$, assuming that the input is of the form $w' = ww$, reads and copies symbols of the input string to the pushdown and guesses when the prefix w has been copied, i.e., when the pushdown content is w^R read from top to down. Until this point the automaton remains in state q_0. Then, without reading any symbol, it pushes \$ to the top of the pushdown, changes from state q_0 to state q_1, and pushes some nondeterministically chosen symbols to the pushdown without reading any symbol from the input. After that phase, $\mathcal{M}$ enters state q_c and checks whether the pushdown content is of the form $w\$w^R$ (without Z_0 at the bottom), i.e., whether or not the pushdown content belongs to R. If this is the case, then $|w\$w^R| = n+1$, i.e., the condition given by f also holds, and the automaton enters state q_2 and compares the string w from the pushdown top against the rest of the input. It enters the final state q_f if and only if the input is of the form ww. It is easy to see that $\mathcal{M}$ accepts exactly the words of L.

The language L can also be accepted by $\mathcal{M}$ working in the weak mode because any word from $L_s(\mathcal{M})$ can be accepted by $\mathcal{M}$ in the weak mode. Furthermore, no more words are in $L_w(\mathcal{M})$ since if the symbol $\#$ appears in the pushdown, then $\mathcal{M}$ cannot enter the final state because Z_0 cannot be the topmost symbol anymore. This means that all the words accepted by $\mathcal{M}$ working in the weak mode are exactly the words of L.

Now we state the following theorem.

Theorem 5.1. $\mathcal{L}_w(\text{PDA}, O(n), \text{LIN}) = \mathcal{L}_s(\text{PDA}, O(n), \text{LIN}) = \text{CS}$.

Proof:
Let $L \subseteq \Sigma^*$ be a context-sensitive language. Then, there exists a linear bounded automaton $\mathcal{K} = (Q_\mathcal{K}, \Sigma, \Gamma_\mathcal{K}, \delta_\mathcal{K}, q_0, q_{acc}, \rhd, \lhd)$ which accepts L. Denote the set of transition rules of $\mathcal{K}$ by $\mathcal{P}$. We first present a blackhole R-sPDA $\mathcal{M} = (\mathcal{M}', Q_c, R)$ with a linear control language R and a linear depth function f such that $L = L_s(\mathcal{M})$ holds.

The basic idea of the proof is that the blackhole R-sPDA $\mathcal{M}$ with input alphabet Σ and alphabet of pushdown symbols $\bar{\Gamma} = \Gamma \cup \{\#\}$, where $\Gamma = \Gamma_\mathcal{K} \cup Q_\mathcal{K} \cup \{\$, \rhd, \lhd\}$ and \$ is a new symbol, simulates $\mathcal{K}$ by reproducing configurations of $\mathcal{K}$ in its pushdown and simulating the transitions of $\mathcal{K}$ by rewriting these configurations.

We first present the control language R. Let $R = L_1 \cup L_2 \cup L_3$, where

$$\begin{aligned} L_1 &= \{\$w^R\$w'\$: w' \vdash_\mathcal{K} w\}\Gamma^*, \\ L_2 &= \{\$\$w\$w^R\$: w \in (\Gamma - \{\$\})^*\}\Gamma^*, \text{ and} \\ L_3 &= \{\$\$\$w^R\$w'\$: w' \vdash_\mathcal{K} w, w = w_1 q_{acc} w_2\}\Gamma^*. \end{aligned}$$

Each L_i, $i = 1,2,3$, serves for checking the correctness of the different phases of the simulation. Since L_1, L_2, and L_3 are linear languages, so R is linear as well. It is easy to see that L_1 can be generated

by the linear grammar $G_1 = (\{S, S', S''\}, \Gamma, P_1, S)$, where

$$P_1 = \{S \to Sx : x \in \Gamma\} \cup \{S \to \$S'\$\} \cup \{S' \to xS'x : x \in \Gamma - \{\$\} - Q_{\mathcal{K}}\}$$
$$\cup \{S' \to \beta^R S''\alpha : \alpha \to \beta \in \mathcal{P}\} \cup \{S'' \to xS''x : x \in \Gamma - \{\$\} - Q_{\mathcal{K}}\} \cup \{S'' \to \$\}.$$

Similarly, L_2 can be generated by the linear grammar $G_2 = (\{S, S'\}, \Gamma, P_2, S)$, where

$$P_2 = \{S \to Sx : x \in \Gamma\} \cup \{S \to \$\$S'\$\} \cup \{S' \to xS'x : x \in \Gamma - \{\$\}\} \cup \{S' \to \$\}.$$

Finally, the linear grammar $G_3 = (\{S, S', S''\}, \Gamma, P_3, S)$ with

$$P_3 = \{S \to Sx : x \in \Gamma\} \cup \{S \to \$\$\$S'\$\} \cup \{S' \to xS'x : x \in \Gamma - \{\$\} - Q_{\mathcal{K}}\}$$
$$\cup \{S' \to \beta^R S''\alpha : \alpha \to \beta \in \mathcal{P}, \alpha = \alpha_1 q_{acc} \alpha_2\} \cup \{S'' \to xS''x : x \in \Gamma - \{\$\} - Q_{\mathcal{K}}\} \cup \{S'' \to \$\}$$

generates L_3.

Now we present the other components of $\mathcal{M}$. Let $Q = \{q_0, q_{1,1}, q_{1,2}, q_{1,3}, q_{c,1}, q_{c,2}, q_{c,3}, q_f\} \cup \{q_i : 1 \leq i \leq 8\}$, where q_f is the sole accepting state, and $Q_c = \{q_{c,1}, q_{c,2}, q_{c,3}\}$. In the following we define the transition function δ. To help the reader understand the construction, we list the transitions in separate groups, according to the phases of the simulation, and explain their work.

The simulation of the start of a computation in $\mathcal{K}$ is realized by the application of the following transitions. For $x, y \in \Sigma$,

$$\begin{aligned}
&\delta(q_0, \lambda, Z_0) = (q_1, \lhd q_0 \$ Z_0), && \delta(q_2, \lambda, \$) = (q_3, \lhd \$), && \delta(q_4, \lambda, \$) = (q_{c,2}, \$),\\
&\delta(q_1, x, \lhd) = (q_1, x \lhd), && \delta(q_3, \lambda, \lhd) = (q_3, x \lhd), && \delta(q_{c,2}, \lambda, \$) = (q_5, \lambda).\\
&\delta(q_1, x, y) = (q_1, xy), && \delta(q_3, \lambda, y) = (q_3, xy), &&\\
&\delta(q_1, \lambda, y) = (q_2, \$ \rhd y), && \delta(q_3, \lambda, y) = (q_4, \$\$q_0 \rhd y), &&
\end{aligned}$$

First, $\mathcal{M}$ pushes the string $\lhd q_0\$$ into the pushdown, enters state q_1, and then it reads the input and meanwhile pushes the symbols of the input word to the pushdown. During this phase, $\mathcal{M}$ is in state q_1. Then, without reading any letter from the input, it adds the string $\$ \rhd$ on the top of the pushdown and enters state q_2. The obtained word is of the form $\$ \rhd w^R \lhd q_0 \$ Z_0$. After that, $\mathcal{M}$ pushes one symbol $\lhd$ into the pushdown and enters state q_3. Then, still in state q_3, it pushes nondeterministically chosen letters into the pushdown and then the string $\$\$q_0 \rhd$, and enters state q_4. Then, without reading any symbol and entering state $q_{c,2}$, it checks whether the resulting pushdown content is a string of the form $\$\$q_0 \rhd w \lhd \$ \rhd w^R \lhd q_0 \$ Z_0$, see the language L_2. Note that by definition, the bottom symbol Z_0 is ignored in the checking. If this is the case, then the simulation of a computation in $\mathcal{K}$ may start; the machine enters state q_5 and removes one $\$$ from the top of the pushdown.

Next, a sequence of transitions in $\mathcal{M}$ follows which simulates a computation in $\mathcal{K}$. The following transitions of $\mathcal{M}$ are for simulating a move in $\mathcal{K}$. For $x, y \in \Gamma - \{\$\}$,

$$\begin{aligned}
&\delta(q_5, \lambda, \$) = (q_6, \$), && \delta(q_{c,1}, \lambda, \$) = (q_7, \$), && \delta(q_8, \lambda, y) = (q_{c,3}, \$\$\$y),\\
&\delta(q_6, \lambda, \$) = (q_6, x\$), && \delta(q_7, \lambda, \$) = (q_8, y\$), && \delta(q_{c,3}, \lambda, \$) = (q_f, \$),\\
&\delta(q_6, \lambda, y) = (q_6, xy), && \delta(q_8, \lambda, y) = (q_8, xy), && \delta(q_{c,2}, \lambda, \$) = (q_5, \lambda).\\
&\delta(q_6, \lambda, y) = (q_{c,1}, \$y), && \delta(q_8, \lambda, y) = (q_{c,2}, \$\$y), &&
\end{aligned}$$

We explain how a configuration change is simulated. Assume that $\mathcal{K}$ performs a move $w' \vdash_{\mathcal{K}} w$. Then, $\mathcal{M}$ simulates this step as follows. Suppose that $\mathcal{M}$ is in state q_5 and its pushdown content is of the form $\$w'\$\gamma Z_0$, for some $\gamma \in \Gamma^*$. Then, $\mathcal{M}$ enters state q_6 and without reading any input symbol it nondeterministically pushes symbols to the pushdown. At some point, $\mathcal{M}$ enters checking state $q_{c,1}$ in which it verifies that the pushdown content is a string from L_1, i.e., whether the obtained pushdown content is of the form $\$w^R\$w'\$\gamma' Z_0$, where w is a successor configuration of w' in $\mathcal{K}$, for some $\gamma' \in \Gamma^*$ such that γ' is a prefix of γ. If this is the case, it enters state q_7, that is, the above step of $\mathcal{K}$ has been simulated on the pushdown in a correct manner; otherwise, $\mathcal{M}$ is not able to continue the computation. It remains to reverse w^R on the top of the pushdown. This is done in a similar way. First, $\mathcal{M}$ enters state q_8 and then it pushes nondeterministically chosen letters from $\Gamma - \{\$\}$ to the pushdown. After a while, it enters the checking state $q_{c,2}$ and it verifies whether or not the resulting pushdown content is a string from L_2. If this is the case, then the pushdown content is $\$\$w\$w^R\$\gamma'' Z_0$, for some $\gamma'' \in \Gamma^*$, and the simulation of the above move of $\mathcal{K}$ is finished; $\mathcal{M}$ enters state q_5 and the simulation of a move of $\mathcal{K}$ may start again. If the pushdown content is not in L_2, then the computation aborts.

To complete the proof, we show how acceptance in $\mathcal{K}$ is simulated in $\mathcal{M}$. Suppose that $\mathcal{M}$ is in state $q_{c,2}$, up to this point a computation in $\mathcal{K}$ is correctly simulated, and the content of the pushdown of $\mathcal{M}$ is of the form $\$w'\$\gamma Z_0$. We guess that $\mathcal{K}$ enters the accepting state by the next move $w' \vdash_{\mathcal{K}} w$. Then, $\mathcal{M}$ simulates this move as above, but instead of performing transition $\delta(q_8, \lambda, y) = (q_{c,2}, \$\$y)$, for $y \in \Gamma - \{\$\}$, it performs transition $\delta(q_8, \lambda, y) = (q_{c,3}, \$\$\$y)$, $y \in \Gamma - \{\$\}$, i.e., it pushes the string $\$\$\$ to the pushdown and enters the checking state $q_{c,3}$. Thus, the obtained word is of the form $\$\$\$w^R\$w'\$\gamma' Z_0$. If the move in $\mathcal{K}$ leads to acceptance, then w^R contains q_{acc}, therefore $\$\$\$w^R\$w'\$\gamma'$ has to be an element of L_3. Thus, $\mathcal{M}$ checks whether or not $\$\$\$w^R\$w'\$\gamma' \in R$ holds, and if this is the case, then enters its final state q_f; otherwise, the computation aborts.

As w is at most of the same length as the input, and we need five \$ signs, four end markers, and two state symbols to represent the configurations in the pushdown, the depth function can be chosen as $f(n) = 2n + 11$.

By the above explanations, the reader may see that $L_s(\mathcal{M}) = L$ holds. Note that the above proof works for $L \in \Sigma^+$. If $\lambda \in L$, then we add the transition $\delta(q_0, \lambda, Z_0) = (q_f, Z_0)$.

The same proof holds for the case of weak mode as well, i.e., $L_w(\mathcal{M}) = L$. This follows from the property that $\mathcal{M}$ never removes any symbol from the pushdown. Thus, any accepting computation of $\mathcal{M}$ performed in the strong mode is also an accepting computation in the weak mode, and no other accepting computation is possible in the weak mode.

On the other hand, consider a blackhole R-sPDA $\mathcal{M} = (\mathcal{M}', Q_c, R, f)$ where R is a linear control language and f is a linear depth function. To see that the language $L_X(\mathcal{M})$, $X \in \{s, w\}$, is context-sensitive, note that $\mathcal{M}$ uses only linear space. Thus, as membership in a linear language can also be checked using linear space by a Turing machine, there is a linear bounded automaton $\mathcal{K}_X$ which simulates $\mathcal{M}$ working in the X-mode, where $X \in \{s, w\}$. □

6. Deterministic Variants

By examining the proof of Theorem 5.1, we can observe that R is linear, deterministic context-free. If the core pushdown automaton of the blackhole R-sPDA is also deterministic, then L is deterministic context-sensitive (in the sense that it can be accepted by a deterministic linear bounded automaton). On

the other hand, if the LBA is deterministic, then R is linear, deterministic context-free. However, the core PDA is nondeterministic in that construction. Thus, as a corollary of Theorem 5.1, we obtain the following statement.

Proposition 6.1. For any deterministic context-sensitive language L, there is a blackhole R-sPDA $\mathcal{M}$ with a linear and deterministic context-free control language, and with a linear depth function such that $L = L_X(\mathcal{M}),\ X \in \{w, s\}$.

We now consider the case of deterministic blackhole R-sPDA, that is, the case when the core PDA of the blackhole automaton is deterministic.

Theorem 6.1. DCF $\subset \mathcal{L}_X(\text{DPDA}, \omega(n), \text{LIN}) \subseteq \mathcal{L}_X(\text{DPDA}, O(n), \text{LIN})$, for $X \in \{w, s\}$.

Proof:
Consider a blackhole state-controlled deterministic PDA $\mathcal{M}'_1 = ((\mathcal{M}_1, f), Q_c, R)$, for $f \in \omega(n)$. It is known [2, Lemma 12.1] that for every DPDA $\mathcal{M}_1$, there is an equivalent DPDA $\mathcal{M}_2$ such that $\mathcal{M}_2$ neither loops infinitely nor its pushdown content grows infinitely while reading no input. Thus, there exists a constant k such that $\mathcal{M}_2$ increases its pushdown height of not more than k symbols while reading no input. In addition, let r denote the maximum of the length of strings that $\mathcal{M}_2$ can push to the pushdown in one step reading the input, i.e., reading the input, the automaton cannot increase the pushdown height of more than $r - 1$ symbols in one step. Considering an input of length n, $\mathcal{M}_2$ can increase the height of its pushdown of not more than $g(n) = n \cdot (r - 1) + (n + 1) \cdot k$ symbols, which is linear with respect to the input length.

Thus, as the automata $\mathcal{M}_1$ and $\mathcal{M}_2$ are equivalent, i.e., accept the same language, and going through a loop containing a checking state is possible only if the pushdown content forms a string belonging to R, the automata $\mathcal{M}'_1 = ((\mathcal{M}_1, f), Q_c, R)$ and $\mathcal{M}'_2 = ((\mathcal{M}_2, g), Q_c, R)$ are equivalent as well. As a consequence, we obtain that for every blackhole R-sDPDA $\mathcal{M}$ with a linear control language, there is a blackhole R-sDPDA $\mathcal{M}'$ whose depth function is linear and $L_X(\mathcal{M}) = L_X(\mathcal{M}')$, $X \in \{w, s\}$, holds. Hence, we have DCF $\subseteq \mathcal{L}_X(\text{DPDA}, \omega(n), \text{LIN}) \subseteq \mathcal{L}_X(\text{DPDA}, O(n), \text{LIN})$, for $X \in \{w, s\}$.

To prove the strictness of the first inclusion, consider the language $\{a^n b^n c^n : n \geq 1\}$ which can be accepted by a blackhole DPDA with a depth function $f \in \omega(n)$.

Construct an R-sDPDA $\mathcal{M} = (\mathcal{M}', Q_c, R)$ such that $\mathcal{M}' = (\mathcal{M}'', f)$ is a blackhole PDA with $\mathcal{M}'' = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{a, b, \$, Z_0\} \cup \{\#\}, \delta, q_0, Z_0, \{q_3\})$, $Q_c = \{q_1\}$ is the set of checking states, and $R = \{b^{n-1}a^n\$: n \geq 1\}$. Let δ be defined as follows. For $x, Z \in \{a, b\}$,

$$\begin{aligned} \delta(q_0, x, Z_0) &= (q_0, x\$Z_0), & \delta(q_0, c, b) &= (q_1, \lambda), & \delta(q_2, c, b) &= (q_2, \lambda), \\ \delta(q_0, x, Z) &= (q_0, xZ), & \delta(q_1, c, b) &= (q_2, \lambda), & \delta(q_2, \lambda, a) &= (q_3, a). \end{aligned}$$

Processing the input, $\mathcal{M}$ pushes into the pushdown a symbol \$ and then all a's and b's that are read. When reading the first c, it removes one b from the pushdown and checks whether the pushdown content is of the form $b^{n-1}a^n\$Z_0$, for some $n \geq 1$. (The control language R is linear, even deterministic context-free.) If this is the case, then the automaton continues processing the input word with comparing the number of c's to the number of b's stored in the pushdown. $\mathcal{M}$ enters the final state q_3 if the number of c's is equal to the number of b's, i.e., after reading the whole input, only letters a remain in the pushdown. Thus, $\mathcal{M}$ accepts any word in the language $\{a^n b^n c^n : n \geq 1\}$, and due to the transitions given above it accepts no other words. □

For blackhole DPDA without control languages, we can prove the following statement.

Theorem 6.2. DCF $= \mathcal{L}_X(\text{DPDA}, \omega(n)) \subset \mathcal{L}_X(\text{DPDA}, O(n))$, for $X \in \{w, s\}$.

Proof:
The fact that DCF $\subseteq \mathcal{L}_X(\text{DPDA}, \omega(n))$, $X \in \{w, s\}$, can be seen as follows. First, for any deterministic context-free language L we can take a DPDA $\mathcal{M}$ accepting L such that $\mathcal{M}$ only uses a portion of the pushdown having a length which is a linear function of the length of the input. Thus, $\mathcal{M}$ also accepts L with any depth function $f \in \omega(n) \subseteq \Omega(n)$. On the other hand, for any blackhole DPDA $\mathcal{M}'$, there is an equivalent blackhole DPDA $\mathcal{M}$ that never uses more than linear space (see the first part of the proof of Theorem 6.1), i.e., any word $w \in L(\mathcal{M})$ of length n can be accepted with pushdown height not greater than $f(n)$, where f is a linear function, and thus the blackhole principle is not applied.

The strictness of the inclusion follows from the proof of Theorem 4.4 since the PDA constructed there to accept the non-context-free language $L = \{a^i b^j c^j : i > j \geq 1\}$ is deterministic. □

We know that any PDA can be considered as a blackhole automaton where the depth of the pushdown is unbounded, i.e., the depth function f is defined as $f(n) = \infty$, for all $n \in \mathbb{N}$. As it is known that every state-controlled R-PDA with a regular control language can effectively be transformed to an equivalent pushdown automaton (see, for example, [4]), we have $\mathcal{L}_X(\text{PDA}, \infty) = \mathcal{L}_X(\text{PDA}, \infty, \text{REG}) = \text{CF}$, for $X \in \{w, s\}$.

In the case of DPDA, we have an analogous statement also for depth functions which are not unbounded, but have a "big enough" bound, that is, for $f \in \omega(n)$.

Theorem 6.3. DCF $= \mathcal{L}_X(\text{DPDA}, \omega(n)) = \mathcal{L}_X(\text{DPDA}, \omega(n), \text{REG})$, for $X \in \{w, s\}$.

Proof:
The first equality is from Theorem 6.2. The second equality can be seen by a construction similar to the one found in [4] for showing that every state-controlled R-PDA with a regular control language can effectively be transformed to an equivalent pushdown automaton. The idea is to take the deterministic finite automaton accepting the control language, and record the sequence of states that it would go through when reading the word which forms the pushdown content together with the pushdown symbols.

Let $\mathcal{M}_1 = (\mathcal{M}_1', Q_c, R)$ such that $R \in \text{REG}$, and $\mathcal{M}_1' = (\mathcal{M}_1'', f)$ be a blackhole DPDA with $f \in \omega(n)$ and $\mathcal{M}_1'' = (Q, \Sigma, \Gamma_1 \cup \{\#\}, \delta_1, q_I, Z_0, F)$ such that it never uses more than linear space (see the first part of the proof of Theorem 6.1). Let $M = (Q_M, \Gamma_1, q_0, \delta_M, F_M)$ be a deterministic finite automaton that accepts the mirror image of the language R, that is, $L(M) = R^R$. Now we construct a blackhole PDA $\mathcal{M}_2 = (\mathcal{M}_2', f)$ such that $L_s(\mathcal{M}_2) = L_s(\mathcal{M}_1)$.

Let the set of pushdown symbols be $\Gamma_2 = \{[r, Z, s] : Z \in \Gamma_1, r, s \in Q_M, \delta_M(s, Z) = r\}$, and let $\mathcal{M}_2' = (Q, \Sigma, \Gamma_2 \cup \{\#\}, \delta_2, q_I, [q_I, Z_0, q_I], F)$. The transition function δ_2 is defined as follows. For all $(q', \gamma) \in \delta_1(q, a, Z)$, $q, q' \in Q$, $q \notin Q_c$, $a \in \Sigma \cup \{\lambda\}$, $Z \in \Gamma_1$, and $\gamma \in \Gamma_1^*$ we have

$$(q', \gamma') \in \delta_2(q, a, Z')$$

where, if $\gamma = Z_1 Z_2 \dots Z_t$ and $Z' = [r, Z, s] \in \Gamma_2$, then $\gamma' = [s_1, Z_1, s_2][s_2, Z_2, s_3] \dots [s_t, Z_t, s]$. If $(q', \gamma) \in \delta_1(q, a, Z)$ and $q \in Q_c$, then we define $\delta_2(q, a, Z')$ as above, but only for $Z' = [r, Z, s]$ where $r \in F_M$.

The blackhole PDA $\mathcal{M}_2$ works by simulating the transitions of $\mathcal{M}_1$, and maintaining not only the same pushdown content, but also the sequence of states that the finite automaton M traverses starting from the bottom of the pushdown and moving up to the top symbol. If the top symbol of the pushdown is $[r, Z, s]$ for some $r, s \in Q_M$ and $Z \in \Gamma_1$, then M is in state r when reading the pushdown content from the bottom up. Therefore, by defining the transitions of $\mathcal{M}_2$ in such a way that from checking states it can only continue working in the case when the top pushdown symbol is of the form $[r, Z, s]$ for some accepting state $r \in F_M$, $\mathcal{M}_2$ is able to simulate the checking functionality of $\mathcal{M}_1$.

Finally, in the case of the weak mode, we add sets of states to the pushdown symbols instead of only states to encode all possible computations starting from any pushdown symbol of the original automaton. Thus, the initial pushdown symbol is of the form $[\{q_I\}, Z_0, \{q_I\}]$ and $\Gamma_2 = \{[X, Z, Y] : Z \in \Gamma_1, X, Y \subseteq Q_M, \delta_M(X, Z) = Y\}$. The transition function δ_2 is then defined so that for all $(q', \gamma) \in \delta_1(q, a, Z)$, $q, q' \in Q$, $q \notin Q_c$, $a \in \Sigma \cup \{\lambda\}$, $Z \in \Gamma_1$, and $\gamma \in \Gamma_1^*$, we have

$$(q', \gamma') \in \delta_2(q, a, Z')$$

where, if $\gamma = Z_1 Z_2 \dots Z_t$ and $Z' = [X, Z, Y] \in \Gamma_2$, then $\gamma' = [Y_1, Z_1, Y_2][Y_2, Z_2, Y_3] \dots [Y_t, Z_t, Y]$, while if $q \in Q_c$, then we define $\delta_2(q, a, Z')$ as above, but only for such $Z' = [X, Z, Y]$ where $X \cap F_M \neq \emptyset$. □

By Theorem 5.1, languages accepted by blackhole DPDA are context-sensitive. The following example demonstrates that deterministic blackhole R-sPDA can accept complicated languages even with regular control language if the depth function is linear.

Example 6.1. Let $L_k = \{(wc)^k : w \in \{a, b\}^+\}$ for $k \geq 1$. Then, for every $k \geq 1$, there is a blackhole R-sDPDA $\mathcal{M}_k$ with a linear depth function f and a regular control language R which in the strong mode accepts L_k. Note that for all $k \geq 2$, L_k are not context-free.

Let $f(n) = \lfloor \frac{n}{k} \rfloor + 1$. We construct $\mathcal{M}_k$ which works as follows. First, $\mathcal{M}_k$ copies the prefix wc of the input to the pushdown. Then, it reads the next symbol, pushes it to the pushdown, and checks whether the first and the last symbols are the same, i.e., the pushdown contains a string from the regular control language $R = \{\ x\{a, b, c\}^* x : x \in \{a, b, c\}\ \}$. According to f, only wcx, for some $x \in \{a, b, c\}$, can be stored in the pushdown. For instance, if $w = ab$, then the pushdown content is $cbaZ_0$. Reading the next input symbol, x, and pushing it to the pushdown, $xcbaZ_0$ is stored. The computation continues if and only if $x = a$. Then, another symbol, say y, is read and pushed to the pushdown, i.e., the pushdown contains $yxcbZ_0$. This implies $y = b$. It means that if the next c is read and pushed to the pushdown, then the string stored in the pushdown will be $cbacZ_0$. The cycle is repeated until the whole input is read. As k is a constant, the number of c's can be counted using the internal states of $\mathcal{M}_k$. It is easy to see that $\mathcal{M}_k$ accepts all words of L_k but no other words.

Note that the power of $\mathcal{M}_k$ is provided by the depth function which is known to the automaton without using any of its computational resources. This is in accordance with the fact that, by Proposition 6.3, there is no blackhole R-sPDA with a regular control language and an unbounded depth function accepting L_k, for $k \geq 2$.

7. Conclusions and Open Problems

In this paper, we introduced and examined blackhole pushdown automata where a symbol can always be pushed to the pushdown, but only a given depth of the pushdown content is considered; the rest of the pushdown content is lost: it is either deleted or replaced by a dedicated symbol $\#$. In the first case the blackhole PDA works in the strong mode, while in the latter case it works in the weak mode. We also studied the corresponding variant of regulated pushdown automata, the so-called blackhole state-controlled R-PDA, which is a blackhole pushdown automaton that in some special states checks the form of its pushdown content according to the membership to a given control language R. We provided the characterization of several language families in terms of these constructs. Among other things, we proved that blackhole state-controlled R-PDA with linear control languages and linear depth functions characterize the context-sensitive language class in any of the working modes.

There have remained several open problems to study. In [5], state-controlled R-PDA with linear control languages were shown to be able to accept any recursively enumerable language checking the pushdown only twice during any computation. Obviously, the same number of checks is needed to obtain this accepting power in the case of blackhole R-sPDA with linear control languages and unbounded depth functions as well. However, it is an open question whether or not for any natural number k, there exists a language that cannot be accepted by a blackhole R-sPDA with a linear control language and a linear depth function checking the pushdown content less than k times.

We have shown that deterministic context-sensitive languages can be accepted by blackhole R-sPDA with linear and deterministic context-free control languages and with a linear depth function. A challenging question is to relate the well-known LBA problem to blackhole R-sPDA.

Acknowledgments

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

References

[1] Csuhaj-Varjú, E., Masopust, T., Vaszil, G.: Blackhole state controlled regulated pushdown automata, *Proceedings of the Second Workshop on Non-Classical Models of Automata and Applications (NCMA 2010)* (H. Bordihn, R. Freund, T. Hinze, M. Holzer, M. Kutrib, F. Otto, Eds.), band 263 of books@ocg.at, Austrian Computer Society, 2010.

[2] Hopcroft, J., Ullman, J.: *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, Massachusetts, 1969.

[3] Křivka, Z.: *Rewriting Systems with Restricted Configurations*, Ph.D. Thesis, Faculty of Information Technology, Brno University of Technology, Brno, 2008.

[4] Kutrib, M., Malcher, A., Werlein, L.: Regulated nondeterminism in pushdown automata, *Theoretical Computer Science*, **410**(37), 2009, 3447–3460.

[5] Masopust, T.: Regulated Nondeterminism in Pushdown Automata: The Non-Regular Case, *Fundamenta Informaticae*, **104**(1-2), 2010, 111–124.

[6] Salomaa, A.: *Formal Languages*, Academic Press, New York, 1973.

[7] Salomaa, A.: *Computation and Automata*, Cambridge University Press, Cambridge, 1985.

[8] Szepietowski, A.: *Turing Machines with Sublogarithmic Space*, vol. 843 of *Lecture Notes in Computer Science*, Springer, Berlin, 1994.