BLACKHOLE STATE-CONTROLLED REGULATED PUSHDOWN AUTOMATA

Erzsébet Csuhaj-Varjú[1,2], Tomáš Masopust[3,4] and György Vaszil[1]

[1]Computer and Automation Research Institute, Hungarian Academy of Sciences
Kende u. 13–17, 1111 Budapest, Hungary
Email: csuhaj@sztaki.hu, vaszil@sztaki.hu
[2]Dep. of Algorithms and Their Applications, Faculty of Informatics, Eötvös Loránd University
Pázmány Péter sétány 1/c, 1117 Budapest, Hungary
[3]Institute of Mathematics, Czech Academy of Sciences
Žižkova 22, 61662 Brno, Czech Republic
Email: masopust@ipm.cz
[4]CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract
In this paper, we introduce and study a variant of regulated pushdown automata, called blackhole state-controlled R-PDA where a symbol can always be pushed to the pushdown, but only a given depth of the pushdown contents is remembered; the rest of the pushdown contents is lost. In addition, the automaton can check the form of its pushdown contents according to a given control language. We present characterizations of several language families in terms of these constructs.

1. Introduction

Recently, regulated pushdown automata have obtained increased interest. In [2], a variant of these accepting computational devices is discussed. There the automata are 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 contents of each step forms a string belonging to the given control language. It is known 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 is also presented showing that non-regular linear control languages increase the computational power of these automata. The question concerning the computational power of these automata with non-regular linear control languages has been examined in [4].

Studying nondeterminism in pushdown automata, the previous modification has been generalized, and so-called R-PDA have been introduced in [3]. Given a control language R, an R-PDA is a pushdown automaton which makes a nondeterministic step whenever the pushdown contents forms a string belonging to R, and makes a deterministic step whenever the pushdown

contents forms a string that does not belong to R, i.e., R-PDA behave nondeterministically if and only if their pushdown contents forms a string belonging to R. It has been shown in [3] that regular control languages do not change the computational power, while non-regular, linear control languages increase the power and make the R-PDA computationally complete [4].

Motivated by R-PDA, which check the form of their pushdown contents in each computational step, so-called state-controlled R-PDA (R-sPDA) have been introduced and studied in [4]. These automata check the form of their pushdown contents only in some special checking states. Again, if R is regular, the computational power is unchanged, while if R is non-regular, linear, the power is that of Turing machines. In addition, two checks of the form of the pushdown contents are sufficient to accept any recursively enumerable language.

Continuing the previous research, in this paper we introduce the concept of a blackhole state-controlled R-PDA (blackhole R-sPDA, for short), where a symbol can always be pushed to the pushdown, but only a given depth of the pushdown contents is considered (the rest of the pushdown contents is lost). Furthermore, in some special states the automaton can check the form of its pushdown contents according to 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 contents are needed or are useful only during a limited time period of the computation, but never later. This implies that this part of the pushdown can be lost or forgotten.

We present the characterization of several language families accepted by blackhole R-sPDA. We prove 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. We also show that every context-free language can be accepted by a blackhole R-sPDA with a regular control language and a linear depth function. But, there exists a non-context-free language that can be accepted by a blackhole R-sPDA with a regular control language and sub-logarithmic depth function. Finally, we raise open problems for future research.

2. Preliminaries and Definitions

In this paper, we assume that the reader is familiar with automata and formal language theory (see, e.g., [5, 6]). For a set A, $|A|$ denotes the cardinality of A. For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V, where the unit is denoted by λ. Set $V^+ = V^* \setminus \{\lambda\}$. For a string $w \in V^*$, the length of w is denoted as $|w|$, and the mirror image of w as 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 is denoted by $\mathbb{N}$.

A *pushdown automaton* (PDA) 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, δ is a transition function from $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ to the set of finite subsets of $Q \times \Gamma^*$, $q_0 \in Q$ is the initial state, $Z_0 \in \Gamma$ is

the initial pushdown symbol, and $F \subseteq Q$ is the set of accepting states. A *configuration* of $\mathcal{M}$ is a triple (q, w, γ), where q is the current state, w is the unread part of the input, and γ is the current contents of the pushdown (the leftmost symbol of γ is the top pushdown symbol). If $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $\gamma, \beta \in \Gamma^*$, $Z \in \Gamma$, and $(p, \beta) \in \delta(q, a, Z)$, then $\mathcal{M}$ makes a move from $(q, aw, Z\gamma)$ to $(p, w, \beta\gamma)$, formally $(q, aw, Z\gamma) \vdash_{\mathcal{M}} (p, w, \beta\gamma)$. For 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 β does not contain Z_0, or $\beta = \beta' Z_0$, where β' does not contain Z_0 and $Z = Z_0$. As usual, the reflexive and transitive closure of the relation $\vdash_{\mathcal{M}}$ is denoted by $\vdash^*_{\mathcal{M}}$ (the subscript $\mathcal{M}$ may be removed if no confusion arises). The *language accepted* by $\mathcal{M}$ is defined as $T(\mathcal{M}) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^*_{\mathcal{M}} (q, \lambda, \gamma) \text{ for some } q \in F \text{ and } \gamma \in \Gamma^*\}$.

A pushdown automaton $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is *deterministic* (DPDA) if there is no more than one move the automaton can make from any configuration, i.e., 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 $a \in \Sigma$. 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 such that $N \cap T = \emptyset$, $S \in N$ 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 of G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$, where $\Rightarrow^*$ is the reflexive and transitive closure of the relation $\Rightarrow$.

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, [5, 6].

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

2.1. Blackhole State-Controlled R-PDA

In the following we define the notion of a blackhole state-controlled regulated pushdown automaton. A quadruple $\mathcal{M} = (\mathcal{M}', Q_c, R, f)$ is called a *blackhole state-controlled* R-PDA (a blackhole R-sPDA, for short), if $\mathcal{M}' = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA, $Q_c \subseteq Q$ is a subset of states called the set of *checking states*, $R \subseteq (\Gamma \setminus Z_0)^*$ is a *control language*, and $f : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ is a function called the *depth function*.

A blackhole R-sPDA $\mathcal{M}$ functions as follows. Let $w_0 \in \Sigma^*$ be an input word, $q, q' \in Q$,

$a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$. Then

$$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \mathtt{trim}_{f(|w_0|)}(\gamma'\gamma))$$

provided that $(q', \gamma') \in \delta(q, a, Z)$ and

1. either $q \in Q \setminus Q_c$,
2. or $q \in Q_c$, $Z\gamma = \gamma'' Z_0$, and $\gamma'' \in R$,

where $\mathtt{trim}_{f(|w_0|)} : \Gamma^* \to \Gamma^*$, for $f(|w_0|) \in \mathbb{N}$, is a function defined as follows:

$$\mathtt{trim}_{f(|w_0|)}(\alpha Z_0) = \begin{cases} \alpha Z_0 & \text{if } |\alpha| \leq f(|w_0|) \\ \alpha' Z_0 & \text{if } |\alpha| > f(|w_0|),\ \alpha = \alpha'\beta,\ |\alpha'| = f(|w_0|),\ \beta \in \Gamma^* \,. \end{cases}$$

If $f(|w_0|) = \infty$, then f is said to be unbounded. In this case $\mathtt{trim}_{f(|w_0|)}(\alpha Z_0) = \alpha Z_0$ for any $\alpha \in \Gamma^*$, i.e., the length of the pushdown contents taken into consideration while $\mathcal{M}$ makes a move is irrelevant, it can be arbitrarily large. This particular case of blackhole R-sPDA coincides with R-sPDA introduced and investigated in [4].

Note that if $q \in Q_c$ and $\gamma'' \notin R$, then $\vdash_{\mathcal{M}}$, called a move of $\mathcal{M}$, is not defined.

Informally, the blackhole state-controlled R-PDA works in such way that it can always push a string to its pushdown, but only the topmost $f(|w_0|)$ symbols are left in the pushdown. The other symbols at positions greater than $f(|w_0|)$ are lost.

Analogously to pushdown automata, the *language accepted* by $\mathcal{M}$ is defined as $T(\mathcal{M}) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^*_{\mathcal{M}} (q, \lambda, \gamma) \text{ for some } q \in F \text{ and } \gamma \in \Gamma^*\}$.

As we mentioned in the Introduction, the main motivation to introduce these variants of automata comes from the investigation of regulated pushdown automata where it turns out that in many cases, certain parts of the pushdown contents are needed or are useful only during a limited time period of the computation, but never later. Therefore, this part of the pushdown can be lost or forgotten. For instance, to accept the language $\{w_1 w_1 w_2 w_2 \dots w_n w_n : w_i \in \{a, b\}^*\}$, there is a blackhole R-sPDA requiring only $\max_{i=1,\dots,n}(2|w_i|+1)$ pushdown cells as demonstrated in Example 9.

3. An Introductory Example

In this example, we consider the case of sub-logarithmic (sub-linear) depth functions.

Example 1. Let $L \subseteq \{a, b\}^*$, and let us consider the language

$$L' = \bigcup_{w \in L} \bigcup_{i=0}^{2^{2^{|w|}}-1} \{a^{(2^{2^{|w|}}+i-(|w|+1))} cw\} \,.$$

We can construct a blackhole R-sPDA $\mathcal{M} = (\mathcal{M}', Q_c, R, f)$ with a sub-logarithmic depth function $f(n) = \lfloor \log\log n \rfloor + 1$ accepting L'.

Note first that any string $w' \in L'$ can be written as $w' = a^k cw$ where if $|w| = l$, then $2^{2^l} - (l+1) \leq k \leq 2^{2^l+1} - 1 - (l+1)$. To check if $w' \in L'$, the automaton reads the input $w' = a^k cw$ until the symbol c, then c and the rest of the input is copied to the pushdown.

Now the automaton performs a pushdown check verifying whether $(cw)^R = w^R c \in L^R\{c\}$ where the presence of c ensures that the prefix consisting of all a's is of sufficient length. To see this, note that the presence of c implies that $f(|w'|) = \lfloor \log\log |w'| \rfloor + 1 \geq |w| + 1$, that is, $\lfloor \log\log(k + l + 1) \rfloor + 1 \geq l + 1$ which means that $k \geq 2^{2^l} - (l + 1)$.

On the other hand, if $w' \in L'$, then $k \leq 2^{2^l+1} - (l+2)$. Since $|w'| = k+l+1$, and $\log\log(k+l+1) \leq \log\log(2^{2^l+1} - 1) < l + 1$, we have that $f(|w'|) = \lfloor \log\log(|w'|) \rfloor + 1 < l + 2$. As $|cw| = l + 1$, this means that the symbol c occupies the last available position on the bottom of the pushdown: if one more symbol is pushed to the pushdown, c disappears from the bottom.

Thus, the automaton can not only check if the prefix of a's is too short by checking the presence of c as described above, but it can also check that the sequence of a's is not too long as follows: if c can be removed by pushing another symbol to the pushdown, say \$, then it accepts. The disappearance of the symbol c can be detected by checking the pushdown contents according to the regular language $\{\$\}\{a, b\}^*$. Summarized, $R = L^R\{c\} \cup \{\$\}\{a, b\}^*$. □

Note that if L is non-recursive, then L' is non-recursive (and also R is non-recursive). Note also that if we set L to be the regular language $\{a, b\}^*$, then L' is not context-free. To see this, assume that L' is context-free. Then, as context-free languages are closed under intersection with regular languages and also under right quotient with regular languages, the language $(L' \cap a^*c(bb)^*)/cb^* = \{a^{(2^{2^n}+i-(n+1))} : n = 2k, k \geq 0, 0 \leq i \leq 2^{2^n+1} - 1\}$ should be context-free, but it is not, as can be shown by using the pumping lemma. Thus, we obtain the following result.

Proposition 2. *There exists a non-context-free language accepted by a blackhole R-sPDA with a regular control language and with depth function $f(n) = o(\log n)$.*

The computational power presented in this proposition comes from the fact that the value of the depth function is known to the automaton without using any of its computational resources. The computation of the depth function, which is necessary for its correct functioning, and which in general could use more workspace than allowed, is not done by the automaton, but it can still use its value during the computation. If, on the other hand, we required that the machine computes the available space itself, that is, we considered a so-called strongly sub-logarithmic space bounded machine according to the terminology of [7], then the computing power would be reduced to the power of finite automata because strongly space-bounded one-way Turing machines accept only regular languages when no more than sub-logarithmic space is used, see [7, Theorem 5.2.1].

4. Regular Control Languages, Constant Depth Functions, and Context-free Languages

State-controlled R-PDA, introduced in [4], can also be considered as blackhole R-sPDA 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, we obtain the following result.

Proposition 3. *If a language L is accepted by a blackhole R-sPDA with a regular control language and an unbounded depth function, then L is context-free.*

If the depth of the pushdown is bounded by a constant, i.e., $f(n) = k$, for some $k \in \mathbb{N}$ and for all $n \in \mathbb{N}$, then the accepted language is regular. To see this, note that a pushdown of constant depth, say k, can have at most a finite number of different configurations, $|\Gamma|^k$ if Γ is the pushdown alphabet. Therefore, if Q is the set of states of the pushdown automaton $\mathcal{M}$, we can construct a finite automaton with the set of states $Q \times \Gamma^k$ which can keep track of the contents of the pushdown, and thus, simulate the functioning of $\mathcal{M}$.

In this case, increasing the complexity of the control language R cannot increase the accepting power of the automaton since only a finite subset consisting of strings of length at most k is of interest, thus, we have the following statement.

Proposition 4. *If a language L is accepted by a blackhole R-sPDA with a recursively enumerable control language and with depth function $f(n) = O(1)$, then L is regular.*

Furthermore, for any context-free language L, there is a context-free grammar in Greibach normal form generating L. As the grammar in Greibach normal form contains no erasing productions and the right-hand side of every rule starts with a terminal symbol, using the standard technique for the construction of pushdown automata based on context-free grammars, we obtain that there is a pushdown automaton $\mathcal{M}$ accepting L requiring not more than linear space.

Proposition 5. *Every context-free language can be accepted by a blackhole R-sPDA with a regular control language and with a linear depth function.*

Moreover, since there exist context-free languages which cannot be accepted by a one-way Turing machine in less than linear space, see [1, Theorem 11.4], it is obvious that there also exist context-free languages which cannot be accepted by any blackhole R-sPDA with a regular control language and a sub-linear depth function $f(n) = o(n)$. We can even formulate the following result.

Proposition 6. *There is a context-free language L, such that no blackhole R-sPDA with an arbitrary recursively enumerable control language can accept L with a depth function $f(n) = o(n)$.*

This can be seen similarly to the proof of [1, Theorem 11.4]. Consider the context-free language $L = \{wcw^R \mid w \in \{a, b\}^*\}$. If L can be accepted by a blackhole R-sPDA $\mathcal{M}$, then after reading the prefix wc of a word wcw^R, $\mathcal{M}$ has to be able to reach 2^l different configurations where $l = |w|$. This is impossible since, 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 not sufficient if f is sub-linear.

5. Linear Control Languages and Linear Depth Functions

As already mentioned in Subsection 2.1, if the depth function is unbounded, then blackhole R-sPDA coincide with state-controlled R-PDA which accept any recursively enumerable language with linear control languages and no more than two checks of the pushdown contents (cf. [4]). Thus, the following statement immediately follows.

Theorem 7. *Let L be a recursively enumerable language. Then, there exists a blackhole R-sPDA $\mathcal{M}$ with a linear control language and with an unbounded depth function such that $L = T(\mathcal{M})$.*

In the sequel, we consider blackhole R-sPDA with linear control languages and with linear depth functions. We show that these automata characterize the family of context-sensitive languages. First, however, two examples demonstrating the main idea of the proof of the statement are presented.

Example 8. 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 accepting L. To do this, let $\mathcal{M} = (\mathcal{M}', Q_c, R, f)$ be defined so that $\mathcal{M}' = (\{q_0, q_1, q_c, q_2, q_f\}, \{a, b\}, \{a, b, \$\}, \delta, q_0, Z_0, \{q_f\})$ is a PDA, $Q_c = \{q_c\}$ is the set of checking states, $R = \{w\$w^R : w \in \{a, b\}^*\}$, and $f(n) = n + 1$ is the (linear) depth function.

Given an input string w', the automaton, assuming that the input is of the form $w' = ww$, nondeterministically copies symbols of the input string to the pushdown and guesses the state when the prefix w has been copied, i.e., the pushdown contents is w^R read from top to down. Then, it pushes \$ to the top of the pushdown, changes from state q_0 to state q_1, and nondeterministically generates (pushes) some symbols to the pushdown. After that phase, it enters state q_c and verifies that the pushdown contents is of the form $w\$w^R$, i.e., it belongs to R. If this is the case, then the automaton enters state q_2 and compares the string w from the pushdown top against the rest of the input. The automaton accepts if and only if the input is of the form ww. □

Repeating the main cycle from Example 8 several times, a similar automaton can accept more complicated languages.

Example 9. As in the previous example and the discussion above, we can accept a language $L_2 = \{w_1 w_1 w_2 w_2 \dots w_k w_k : w_i \in \{a, b\}^*, i = 1, 2, \dots, k,\ k \geq 1\}$ by a blackhole R-sPDA. In this case, the maximal depth of the pushdown which is considered during the computation is not more than $\max_{i=1,2,\dots,k}(2|w_i| + 1)$. □

Before stating the next theorem, we need the notion of a *linear bounded automaton* (an LBA) which is a construct $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, \rhd, \lhd)$ 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 $\rhd, \lhd \in \Sigma$ 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. In accordance with the definition, however, we can assume that the LBA uses a linear number of cells with respect to the length of the input since one symbol can represent the contents of more than one cell of the tape.

A *configuration* of $\mathcal{M}$ is a string of the form w_1qw_2, where the current contents of the tape is represented by $w_1w_2 \in \{\rhd\}(\Gamma^* \setminus \{\lhd, \rhd\})^*\{\lhd\}$, and $q \in Q$ is the current state of the machine. Given an input string w_0, the initial configuration is $q_0\rhd w_0\lhd$. The transition function δ can be rewritten so that if $(q, b, L) \in \delta(p, a)$, then we write $xpa \to qxb$, for all $x \in \Sigma$, 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_1xpaw_2 and a rule $xpa \to qxb$, we define the computational step relation $\vdash$ so that $w_1xpaw_2 \vdash w_1qxbw_2$. Similarly, for a rule $pa \to qp$, we obtain $w_1xpaw_2 \vdash w_1xbqw_2$.

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 $q_0\rhd w_0\lhd$, u_i are configurations for all $i = 1, 2, \ldots, n-1$, and u_n is a configuration of the form $w_1q_{acc}w_2$ containing the accepting state. 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 (see, for example, [5, 6])).

Theorem 10. *A language L is context-sensitive if and only if there exist a blackhole R-sPDA $\mathcal{M}$ with a linear control language and with a linear depth function such that $L = T(\mathcal{M})$ holds.*

Proof. Let $L \subseteq \Sigma^*$ be a context-sensitive language. Then, there exists a linear bounded automaton $\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, \rhd, \lhd)$ accepting L. We present a blackhole R-sPDA $\mathcal{M} = (\mathcal{M}', Q_c, R, f)$ with a linear control language R and a linear depth function f such that $L = T(\mathcal{M})$ holds.

We first construct the control language R. Let $G_1 = (\{S, S', S''\}, \Sigma, P, S)$ be a linear grammar, where the production set P contains productions of the form $S \to Sx$, $S \to \$S'\$$, $S' \to xS'x$, $S'' \to xS''x$, $S'' \to \$$, for all $x \in \Sigma$, and for each transition rule of the form $\alpha \to \beta$ of the LBA $\mathcal{K}$, a production $S' \to \beta^R S''\alpha$. Then, the language generated by G_1, denoted as L_1, is $L_1 = \{\$w^R\$w'\$: w' \vdash_{\mathcal{K}} w\}\Sigma^*$. Similarly, we construct a linear grammar G_2 so that its language is $L_2 = \{\$\$w\$w^R\$: w \in \Sigma^*\}\Sigma^*$, and a linear grammar G_3 generating $L_3 = \{\$\$\$w^R\$w'\$: w' \vdash_{\mathcal{K}} w, w = w_1q_{acc}w_2\}\Sigma^*$. We set $R = L_1 \cup L_2 \cup L_3$ which is the linear control language of $\mathcal{M}$.

The blackhole R-sPDA $\mathcal{M}$ with input alphabet Σ and tape alphabet $\Gamma \cup Q \cup \{\$, \rhd, \lhd\}$ simulates

$\mathcal{K}$ by reproducing configurations of $\mathcal{K}$ in its pushdown and simulating the transitions of $\mathcal{K}$ by rewriting these configurations. In the following we describe the functioning of $\mathcal{M}$, the precise construction of the state and transition sets are left to the reader.

We first show how a move in $\mathcal{K}$ is simulated in $\mathcal{M}$. The simulation of a configuration change is done in several steps. Assume that the LBA $\mathcal{K}$ performs a move $w' \vdash_{\mathcal{K}} w$. Then, $\mathcal{M}$ simulates this step as follows. Assume that $\mathcal{M}$ is in some state q and its pushdown contents is of the form

$$\$w'\$\gamma$$

for some $\gamma \in \Gamma^*$. Then $\mathcal{M}$, without reading any input symbol, nondeterministically pushes symbols to the pushdown, and then enters a checking state in which it verifies that the pushdown contents forms a string from L_1, i.e., whether or not the obtained pushdown contents is of the form

$$\$w^R\$w'\$\gamma'$$

for some $\gamma' \in \Gamma^*$ such that γ' is a prefix of γ. If this is the case, then 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 similar way, i.e., by a nondeterministic push of symbols to the pushdown and checking whether or not the resulting pushdown contents forms a string from R; more specifically, from L_2. If it does, then the pushdown contents is

$$\$\$w\$w^R\$\gamma'',$$

and thus the simulation of the above move of $\mathcal{K}$ is finished.

The simulation of the start of a computation in $\mathcal{K}$ is as follows. First, $\mathcal{M}$ pushes one symbol \$ into the pushdown, then two symbols q_0,$\lhd$, and then it reads the input and meanwhile copies the letters of the input word to the pushdown. Finally it adds a symbol $\rhd$ and one more \$ as the topmost symbol. The obtained word is of the form $\$\rhd w^R \lhd q_0\$Z_0$. Then, as above, $\mathcal{M}$ nondeterministically pushes symbols to the pushdown and checks whether or not the resulting pushdown contents forms a string of the form $\$\$q_0\rhd w\lhd\$\rhd w^R\lhd q_0\$Z_0$. If it is the case, then the simulation of a computation in $\mathcal{K}$ may start.

To complete the proof, we show how acceptance in $\mathcal{K}$ is simulated in $\mathcal{M}$. Suppose that $\mathcal{M}$ is in some state q, up to this point a computation in $\mathcal{K}$ is correctly simulated, and the contents of the pushdown of $\mathcal{M}$ is of the form $\$w'\$\gamma$. 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 written above, but instead of introducing \$ as the topmost symbol, it pushes the symbols \$\$\$ to the pushdown. Thus, the obtained word is of the form $\$\$\$w^R\$w'\$\gamma'$. 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 $\$\$\$w^R\$w'\$\gamma' \in R$ (i.e., L_3) holds, and if this is the case, then enters its final state.

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, $f(n) = 2n + 5 + 4 + 2$ is a suitable depth function.

Consider now 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 $T(\mathcal{M})$ 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 an LBA which simulates $\mathcal{M}$. □

Note that if the core pushdown automaton of the blackhole R-sPDA is deterministic and R is linear, deterministic context-free, then L is deterministic context-sensitive. On the other hand, if the LBA is deterministic, then R is linear, deterministic context-free. Thus, we have the following corollary.

Corollary 11. *Let L be a deterministic context-sensitive language. Then, 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 $L = T(\mathcal{M})$.*

Note, however, that the core PDA of $\mathcal{M}$ is nondeterministic in this construction.

5.1. Deterministic Blackhole R-sPDA

A blackhole R-sPDA $\mathcal{M} = (\mathcal{M}', Q_c, R, f)$ is *deterministic* if $\mathcal{M}'$ is a deterministic pushdown automaton.

It is known [1, 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 contents grows infinitely while reading no input. Thus, there exists a constant k such that $\mathcal{M}_2$ increases its pushdown of not more than k symbols while reading no input. In addition, let r denote the maximal length of strings $\mathcal{M}_2$ can push to the pushdown in one step reading the input, i.e., reading the input, the automaton cannot increase the pushdown of more than $r-1$ symbols in one step. Considering an input of length n, $\mathcal{M}_2$ can increase the length of its pushdown of no more than $n \cdot (r-1) + (n+1) \cdot k$ symbols, which is linear with respect to the input.

Moreover, 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 contents forms a string from R, the automata $(\mathcal{M}_1, Q_c, R, f)$ and $(\mathcal{M}_2, Q_c, R, f)$ are equivalent as well. As a consequence, we obtain the following statement.

Proposition 12. *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 $T(\mathcal{M}) = T(\mathcal{M}')$ holds.*

Obviously, by Theorem 10, languages accepted by these automata are context-sensitive.

The following examples demonstrate blackhole R-sDPDA with linear control languages and a linear depth functions accepting non-context-free languages. For instance, the language $\{a^n b^n c^n : n \geq 1\}$ can be accepted by such an automaton. The automaton stores all a's and b's

read. When reading the first c, it checks that the pushdown contents is of the form $b^n a^n$, for some $n \geq 1$, which is linear (even deterministic context-free), i.e., $R = \{b^n a^n : n \geq 1\}$. If so, the number of c's is compared with the number of b's stored in the pushdown.

In the following example, more complicated languages of deterministic blackhole R-sPDA with regular control languages and linear depth functions are presented.

Example 13. 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 accepting L_k. Note that for all $k \geq 2$, L_k are not context-free.

Let $f(n) = \frac{n}{k} + 1$. 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 that 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 $w = abc$, then the pushdown contents is cba. Reading the next input symbol, x, and pushing it to the pushdown, $xcba$ 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 $yxcb$. 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 $cbac$. 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 $\mathcal{M}_k$. □

Note that similarly to Example 1, 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 3, there is no blackhole R-sPDA with a regular control language and an unbounded depth function accepting L_k, for $k \geq 2$.

6. Conclusions and Open Problems

In this paper, we introduced and examined a variant of regulated pushdown automata, the so-called blackhole state-controlled R-PDA where a symbol can always be pushed to the pushdown, but only a given depth of the pushdown contents is considered; the rest of the pushdown contents is lost. In addition, in some special states, the automaton can check the form of its pushdown contents according to a given control language. We provided the characterization of several language families, we proved that these constructs with linear control languages and linear depth functions characterize the context-sensitive language class.

There have remained several open problems to study. In [4], 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 linear control language and linear depth function checking the pushdown contents 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.

Another important problem area concerns the accepting power of blackhole R-sPDA with linear control languages and sub-linear depth functions.

Acknowledgments

T. Masopust has been supported by the Czech Academy of Sciences, Institutional Research Plan no. AV0Z10190503.

References

[1] HOPCROFT, J.E., ULLMAN, J.D., Formal Languages and Their Relation to Automata, Addison-Wesley, Reading, Massachusetts 1969.

[2] KŘIVKA, Z., Rewriting Systems with Restricted Configurations, PhD thesis, Faculty of Information Technology, Brno University of Technology, Brno 2008.

[3] KUTRIB, M., MALCHER, A., WERLEIN, L., Regulated nondeterminism in pushdown automata. Theoretical Computer Science 410 (37) (2009), 3447–3460.

[4] MASOPUST, T., Regulated nondeterminism in PDAs: The non-regular case, in H. Bordihn, R. Freund, M. Holzer, M. Kutrib, F. Otto (Eds.), Proceedings of the Workshop on Non-Classical Models of Automata and Applications (NCMA 2009), books@ocg.at, band 256, Austrian Computer Society, 2009, 181–194.

[5] SALOMAA, A., Formal Languages, Academic Press, New York 1973.

[6] SALOMAA, A., Computation and Automata, Cambridge University Press, Cambridge 1985.

[7] SZEPIETOWSKI, A., Turing Machines with Sublogarithmic Space, Springer, Berlin 1994.