Fundamenta Informaticae XXI (2001) 1001–1013 1001
IOS Press

Regulated Nondeterminism in Pushdown Automata: The Non-Regular Case

Tomáš Masopust
Mathematical Institute, Czech Academy of Sciences
Žižkova 22, 616 62 Brno, Czech Republic
masopust@math.cas.cz

Abstract. We continue the investigation of pushdown automata which are allowed to make a nondeterministic decision if and only if their pushdown content forms a string belonging to a given control language. We prove that if the control language is linear and non-regular, then the power of pushdown automata regulated in this way is increased to the power of Turing machines. From a practical point of view, however, it is inefficient to check the form of the pushdown content in each computational step. Therefore, we prove that only two checks of the pushdown content are of interest for these machines to be computationally complete. Based on this observation, we introduce and discuss a new model of regulated pushdown automata.

1. Introduction

While finite automata are of great interest in the theory and applications of regular expressions and languages, pushdown automata (PDAs) play an important role in the analysis of programming and natural languages. However, it is well-known that both programming and natural languages have some features that are not context-free, which means that not all these languages can be recognized by pushdown automata. For that reason, there are attempts to introduce some regulating mechanisms to increase the computational power of pushdown automata so that they are able to handle these features without loss of the practical efficiency.

Motivated by some restrictions of context-free derivations studied in regulated rewriting (cf. [3, 4]), so-called regulated pushdown automata have been introduced and studied in [7]. These automata are pushdown automata with an additional control language over the alphabet of transitions restricting the applications of the transition function. An input string is accepted by such a machine whenever the pushdown automaton accepts the input by a sequence of transitions that forms a string belonging to the

Address for correspondence: Mathematical Institute, Czech Academy of Sciences, Žižkova 22, 616 62 Brno, Czech Republic

given control language. On one hand, it has been shown that regular control languages do not affect the power of pushdown automata. On the other hand, regulated pushdown automata with non-regular, linear control languages are computationally complete (the reader is also referred to [10]).

Another variant of pushdown automata with some type of regulation is mentioned in [8], where instead of a control language over the alphabet of transitions the automata are given a control language over the alphabet of pushdown symbols. An input string is accepted whenever the pushdown automaton accepts it by a computation each pushdown content of which forms a string belonging to the given control language. It is proved that if the control language is regular, then the computational power is the same as the power of pushdown automata. On the other hand, an example showing that non-regular, linear control languages increase the power of these machines is presented. Nevertheless, the precise computational power of these machines with non-regular, linear control languages was left open.

Recently, investigating the effect of nondeterminism on computations and the computational power of pushdown automata, the above mentioned modification has been generalized, and so-called R-PDAs have been introduced and studied in [9]. Specifically, given a control language R, an R-PDA is a pushdown automaton which makes a nondeterministic step whenever the pushdown content forms a string that belongs to R, and makes a deterministic step whenever the pushdown content forms a string that does not belong to R. Thus, according to this restriction, the R-PDA behaves nondeterministically if and only if the pushdown content forms a string that belongs to R, and, thus, the nondeterministic behavior of this machine is regulated. It has been shown (see [9]) that regular control languages do not affect the computational power of pushdown automata, while non-regular, linear control languages increase their computational power. For further results and properties concerning R-PDAs, where R is a regular control language, the reader is referred to [9]. In there the case of the precise computational power of R-PDAs with non-regular control languages is formulated as an open problem.

In this paper, we answer this question by showing that R-PDAs are computationally complete even if the control language R is a very simple non-regular language, i.e., a linear language. In addition, from the computational and descriptional complexity viewpoint, we demonstrate that only two checks of the form of the pushdown content are of some interest during any computation and that the number of states and pushdown symbols can be bounded.

Naturally, from the point of view of practical applications, to check the form of the pushdown content in each computational step is not very effective. Therefore, based on the observation that only two checks of the pushdown content are of interest during any computation, we introduce and discuss a new variant of these machines, so-called *state-controlled* R-PDAs (R-sPDAs), which check the form of the pushdown content only in some special states. Specifically, given a control language R, an R-sPDA is a pushdown automaton which has a special set of distinguished states (so-called *checking states*) in which the machine makes a computational step according to its transition function if and only if the pushdown content forms a string that belongs to R; note that if the pushdown content does not form a string from R, the computational process is finished and the machine rejects the input. In all other states, the automaton behaves as an ordinary pushdown automaton. As a result, we have that two checks of the form of the pushdown content make R-sPDAs computationally complete. On the other hand, we show that R-sPDAs with only one check of the pushdown content are more powerful than ordinary pushdown automata. However, their precise computational power is an open problem.

Finally, we discuss R-PDAs and R-sPDAs where the core pushdown automata are deterministic and the control language R is linear (and deterministic context-free), and formulate some open problems.

2. Preliminaries and Definitions

We assume that the reader is familiar with automata and formal language theory (see [12, 13]). 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 of V^* is denoted by ε. Set $V^+ = V^* \setminus \{\varepsilon\}$. For a string $w \in V^*$, $|w|$ denotes the length of w, and w^R denotes the mirror image (or reversal) of w. For a language $L \subseteq V^*$, $L^R = \{w^R : w \in L\}$ denotes the mirror image of L.

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

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 $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ 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 of $\mathcal{M}$, w is the unread part of the input, and γ is the current content of the pushdown (the leftmost symbol of γ is the topmost pushdown symbol). If $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$, $\gamma, \beta \in \Gamma^*$, $Z \in \Gamma$, 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 Z_0 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 *language accepted* by $\mathcal{M}$ is defined as $T(\mathcal{M}) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^*_{\mathcal{M}} (q, \varepsilon, \gamma) \text{ for some } q \in F \text{ and } \gamma \in \Gamma^*\}$.

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 hold:

1. $|\delta(q, a, Z)| \leq 1$, for all $a \in \Sigma \cup \{\varepsilon\}$, $q \in Q$, and $Z \in \Gamma$, and
2. for all $q \in Q$ and $Z \in \Gamma$, if $\delta(q, \varepsilon, Z) \neq \emptyset$, then $\delta(q, a, Z) = \emptyset$, for all $a \in \Sigma$.

In this case, we write $\delta(q, a, Z) = (p, \gamma)$ instead of $\delta(q, a, Z) = \{(p, \gamma)\}$.

Let the family of languages accepted by automata of type X be denoted by $\mathscr{L}(X)$. Then it is well-known that $\mathscr{L}(\text{DPDA}) \subset \mathscr{L}(\text{PDA})$.

2.1. Pushdown Automata with Regulated Nondeterminism

In comparison with the ordinary pushdown automata, R-PDAs are given a control language R over the alphabet of pushdown symbols which restricts the nondeterministic behavior of the machine so that the nondeterministic steps are allowed if and only if the current content of the pushdown forms a string that belongs to R. If it does not belong to R, only deterministic steps are allowed.

Formally, let $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown automaton, and let $R \subseteq (\Gamma \setminus \{Z_0\})^*$ be a control language over the alphabet of pushdown symbols. Then $\mathcal{M}$ is a (bottom-up) R-PDA if the following two conditions are satisfied:

1. for all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $Z \in \Gamma$, δ can be written as

$$\delta(q, a, Z) = \delta_d(q, a, Z) \cup \delta_{nd}(q, a, Z)\,,$$

where $(Q, \Sigma, \Gamma, \delta_d, q_0, Z_0, F)$ is a DPDA and $(Q, \Sigma, \Gamma, \delta_{nd}, q_0, Z_0, F)$ is a PDA, and

2. for all $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

$$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma) \text{ if}$$

   (a) either $(q', \gamma') \in \delta_{nd}(q, a, Z)$, $Z\gamma = \gamma''Z_0$, and $(\gamma'')^R \in R$,
   (b) or $\delta_d(q, a, Z) = (q', \gamma')$, $Z\gamma = \gamma''Z_0$, and $(\gamma'')^R \notin R$.

Condition 2 says that whenever the pushdown content forms a string that does not belong to R, the automaton operates deterministically. Note that these machines check the form of the pushdown content in each computational step, and that this check is made in the *bottom-up* reading direction of the pushdown content.

Analogously, the pushdown content can be checked in the reverse direction, which defines so-called *top-down* R-PDAs. In this case, Condition 2 is replaced with Condition 2' below:

2'. for all $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

$$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma) \text{ if}$$

   (a) either $(q', \gamma') \in \delta_{nd}(q, a, Z)$, $Z\gamma = \gamma''Z_0$, and $\gamma'' \in R$,
   (b) or $\delta_d(q, a, Z) = (q', \gamma')$, $Z\gamma = \gamma''Z_0$, and $\gamma'' \notin R$.

Thus, with respect to the direction in which the pushdown content is read during the check of its form we have two variants of R-PDAs, namely bottom-up and top-down R-PDAs.

3. Computational Power of R-PDAs

In this section, we present the main results of this paper. First, recall that it is known that if the control language R is regular, then every bottom-up R-PDA can effectively be transformed to an equivalent pushdown automaton. In the following, we extend this theorem to top-down R-PDAs.

Theorem 3.1. Let R be a regular control language and $\mathcal{M}$ be a bottom-up or top-down R-PDA. Then an equivalent pushdown automaton $\mathcal{M}'$ can effectively be constructed.

Proof:
For bottom-up R-PDAs, a proof is given in [9]. The case of top-down R-PDAs then follows from the closure property of regular languages under mirror image. □

On the other hand, it has been demonstrated (cf. [8, 9]) that if the control language R is both linear and deterministic context-free, then there is a bottom-up R-PDA accepting a non-context-free language. For top-down R-PDAs, this is also demonstrated in the following example.

Example 3.1. Let $R = \{a^n b^n : n \geq 1\}$ be a language, and R^R its mirror image; R and R^R are both linear and deterministic context-free. Let $\mathcal{M} = (\{q_a, q_b, q_c, q_d, q_f\}, \{a, b, c, d\}, \{a, b, Z_0\}, \delta, q_a, Z_0, \{q_f\})$ be a bottom-up R-PDA (top-down R^R-PDA) operating as follows:

1. starting in q_a, $\mathcal{M}$ deterministically repeats reading a from the input and pushing a to the pushdown;
2. reading the first b, $\mathcal{M}$ deterministically goes to state q_b and pushes b to the pushdown, i.e., the pushdown contains ba^nZ_0; being in q_b, $\mathcal{M}$ deterministically repeats reading b from the input and pushing b to the pushdown;
3. reading the first c, $\mathcal{M}$ goes to state q_c by transitions which belong to δ_{nd}, checking that the pushdown content is $b^na^nZ_0$, and removes b from the top of the pushdown;
4. being in q_c, $\mathcal{M}$ deterministically repeats reading c from the input and removing b from the pushdown;
5. being in q_c and having a on the top of the pushdown, $\mathcal{M}$ deterministically goes to state q_d, reads d from the input, and removes a from the pushdown, i.e., c^n has been read;
6. being in q_d, $\mathcal{M}$ deterministically repeats reading d from the input and removing a from the pushdown;
7. finally, being in q_d and having Z_0 on the top of the pushdown, $\mathcal{M}$ deterministically goes to the final state q_f from which no other symbol can be read; moreover, nothing is read from the input, and Z_0 is removed from the pushdown.

The language recognized by the bottom-up $\{a^nb^n : n \geq 1\}$-PDA (top-down $\{b^na^n : n \geq 1\}$-PDA) $\mathcal{M}$ is $T(\mathcal{M}) = \{a^nb^nc^nd^n : n \geq 1\}$, which is a non-context-free language.

In what follows, we show that every recursively enumerable (RE) language is accepted by a bottom-up (top-down) R-PDA $\mathcal{M}$, for some convenient non-regular, linear control language R. Moreover, in the case of top-down R-PDAs, we show that R can be both linear and deterministic context-free, which is open for bottom-up R-PDAs. Furthermore, we prove some descriptional complexity results.

Theorem 3.2. Let L be an RE language. Then there exist a linear control language R and a bottom-up R-PDA $\mathcal{M}$ such that $L = T(\mathcal{M})$.

To prove this theorem, we need the following Geffert normal form. Let $L \subseteq T^*$ be an RE language. Then, by [5], there is a grammar $G = (\{S, A, B\}, T, P \cup \{ABBBA \to \varepsilon\}, S)$ in Geffert normal form such that $L = L(G)$ and P contains only context-free productions of the following three forms:

$$S \to uSa,\ S \to uSv,\ S \to uv\,,$$

where $u \in \{AB, ABB\}^*$, $v \in \{BBA, BA\}^*$, and $a \in T$. In addition, any successful derivation of G can be divided into the following two parts: the first part is of the form

$$S \Rightarrow_G^* w_1'Sw_2'w \Rightarrow_G w_1w_2w\,,$$

generated only by context-free productions from P, where $w_1 \in \{AB, ABB\}^*$, $w_2 \in \{BBA, BA\}^*$, and $w \in T^*$, and the other part is of the form

$$w_1w_2w \Rightarrow_G^* w\,,$$

generated only by the erasing production $ABBBA \to \varepsilon$. Note also that during the derivation, there is no more than one occurrence of the string $ABBBA$ in w_1w_2, which is "in the middle" of this string.

Proof:
Let $L \subseteq T^*$ be an RE language, and let $G = (\{S, A, B\}, T, P \cup \{ABBBA \to \varepsilon\}, S)$ be a grammar in Geffert normal form such that $L = L(G)$. Let $G_1 = (N_1, T_1, P_1, S_1)$ be a linear grammar, where $N_1 = \{S_1, S\}$, $T_1 = T \cup \{A, B, \$\}$, for \$ and S_1 being new symbols, and $P_1 = P \cup \{S_1 \to \$\$S\}$. Then $L(G_1) = \{\$\$w_1w_2w : S \Rightarrow_G^* w_1'Sw_2'w \Rightarrow_G w_1w_2w\}$. Let $G_2 = (N_2, T_2, P_2, S_2)$ be a linear grammar, where $N_2 = \{S_2, Y, Z\}$, $T_2 = T \cup \{A, B, \$\}$, and $P_2 = \{S_2 \to \$Y, Y \to Z, Z \to ABZBBA, Z \to ABBZBA, Z \to \varepsilon\} \cup \{Y \to Ya : a \in T\}$. Then $L(G_2) = \{\$w_1w_2w : w_1 \in \{AB, ABB\}^*, w_2 \in \{BBA, BA\}^*, w \in T^*\}$, where each $\$w_1w_2w \in L(G_2)$ can be reduced to $\$w$ by the repeated elimination of the string $ABBBA$. In other words, $w_1w_2 \Rightarrow^* \varepsilon$ by the production $ABBBA \to \varepsilon$. Let the linear control language be

$$R = L(G_1)^R \cup L(G_2)^R \cup (\{A, B\} \cup T)^*,$$

and define the R-PDA $\mathcal{M} = (\{q_0, q_1, q_f\}, T, T \cup \{A, B, \$, Z_0\}, \delta, q_0, Z_0, \{q_f\})$, where δ is defined as follows:

$$\begin{aligned}
\delta_{nd}(q_0, \varepsilon, X) &= \{(q_0, aX) : a \in T \cup \{A, B\}\},\ X \in \{A, B, Z_0\} \cup T,\\
\delta_{nd}(q_0, \varepsilon, X) &= \{(q_0, \$\$X)\},\ X \in \{A, B, Z_0\} \cup T,\\
\delta_{nd}(q_0, \varepsilon, \$) &= \{(q_1, \varepsilon)\},\\
\delta_{nd}(q_1, \varepsilon, \$) &= \{(q_1, \varepsilon)\},\\
\delta_{nd}(q_1, \varepsilon, X) &= \{(q_1, \varepsilon)\},\ X \in \{A, B\},\\
\delta_{nd}(q_1, a, a) &= \{(q_1, \varepsilon)\},\ a \in T,\\
\delta_{nd}(q_1, \varepsilon, Z_0) &= \{(q_f, \varepsilon)\}.
\end{aligned}$$

Finally, δ_d is empty.

Informally, $\mathcal{M}$ operates so that it first nondeterministically pushes symbols from $T \cup \{A, B\}$ onto its pushdown, which is possible because $(\{A, B\} \cup T)^* \subseteq R$. Then, when \$\$ is pushed onto the pushdown, i.e., the configuration is of the form $(q_0, w, \$\$\gamma Z_0)$, for some $\gamma \in \Gamma^*$, $\mathcal{M}$ verifies that $\$\$\gamma$ belongs to $L(G_1)$. If so, one symbol \$ is removed, $(q_0, w, \$\$\gamma Z_0) \vdash_{\mathcal{M}} (q_1, w, \$\gamma Z_0)$, and $\mathcal{M}$ verifies that $\$\gamma$ belongs to $L(G_2)$. If so, then $\gamma = w_1w_2w$, where

- $S \Rightarrow^* w_1'Sw_2'w \Rightarrow w_1w_2w$ in G,
- $w_1 \in \{AB, ABB\}^*$, $w_2 \in \{BBA, BA\}^*$, $w \in T^*$, and
- the production $ABBBA \to \varepsilon$ can be used to eliminate the string w_1w_2,

i.e., there is a derivation $w_1w_2w \Rightarrow_G^* w$ in G. The automaton then finishes the computation as follows: $(q_1, w, \$w_1w_2wZ_0) \vdash_{\mathcal{M}} (q_1, w, w_1w_2wZ_0) \vdash_{\mathcal{M}}^* (q_1, w, wZ_0) \vdash_{\mathcal{M}}^* (q_1, \varepsilon, Z_0) \vdash_{\mathcal{M}} (q_f, \varepsilon, \varepsilon)$.

Formally, to prove that $L(G) \subseteq T(\mathcal{M})$, let $S \Rightarrow^* w_1'Sw_2'w \Rightarrow w_1w_2w \Rightarrow^* w$ be a successful derivation of G. Then the corresponding computation of $\mathcal{M}$ accepting w is as follows:

$$\begin{array}{lllllll}
(q_0, w, Z_0) & \vdash^* & (q_0, w, wZ_0) & \vdash^* & (q_0, w, w_2wZ_0) & \vdash^* & (q_0, w, w_1w_2wZ_0)\\
& \vdash & (q_0, w, \$\$w_1w_2wZ_0) & \vdash & (q_1, w, \$w_1w_2wZ_0) & \vdash & (q_1, w, w_1w_2wZ_0)\\
& \vdash^* & (q_1, w, wZ_0) & \vdash^* & (q_1, \varepsilon, Z_0) & \vdash & (q_f, \varepsilon, \varepsilon).
\end{array}$$

Thus, $w \in T(\mathcal{M})$ is satisfied.

On the other hand, to prove $T(\mathcal{M}) \subseteq L(G)$, consider a computation of $\mathcal{M}$ accepting w. Such a computation is of the form

$$\begin{aligned}(q_0, w, Z_0) &\vdash^* (q_0, w, \gamma Z_0)\\ &\vdash (q_0, w, \$\$\gamma Z_0) \quad (1)\\ &\vdash (q_1, w, \$\gamma Z_0) \quad (2)\\ &\vdash (q_1, w, \gamma Z_0) \vdash^* (q_1, \varepsilon, Z_0) \vdash (q_f, \varepsilon, \varepsilon)\end{aligned}$$

for some $\gamma \in \Gamma^*$. From the verification process made during the computational step (2), it follows that $\$\gamma \in L(G_2)$, which means that $\gamma = w_1 w_2 w'$, where $w_1 \in \{AB, ABB\}^*$, $w_2 \in \{BBA, BA\}^*$, $w' \in T^*$, and the string $w_1 w_2$ can be eliminated by the repeated application of the production $ABBBA \to \varepsilon$. Moreover, from the verification process made during the computational step (1), it follows that there is a derivation $S \Rightarrow^* w_1' S w_2' w' \Rightarrow w_1 w_2 w'$ in G. It remains to prove that $w' = w$. However, by examining the following part of the computation,

$$\begin{aligned}(q_1, w, \$w_1 w_2 w' Z_0) &\vdash (q_1, w, w_1 w_2 w' Z_0) \vdash^* (q_1, w, w_2 w' Z_0)\\ &\vdash^* (q_1, w, w' Z_0) \vdash^* (q_1, \varepsilon, Z_0) \vdash (q_f, \varepsilon, \varepsilon)\,,\end{aligned}$$

it immediately follows that the strings w' and w are equal because the only transitions reading the input are of the form $\delta_{nd}(q_1, a, a) = \{(q_1, \varepsilon)\}$, for $a \in T$. Thus, $w \in L(G)$ is satisfied. □

From the descriptional complexity point of view, we have the following corollary.

Corollary 3.1. Let L be an RE language. Then there exist a linear language R and a bottom-up (top-down) R-PDA $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ such that $|Q| \leq 3$, $|\Gamma| \leq |\Sigma| + 4$, and $L = T(\mathcal{M})$.

Proof:
For the top-down R-PDAs, let $R = L(G_1) \cup L(G_2) \cup (\{A, B\} \cup T)^*$. The proof then immediately follows from the construction given in the proof of the previous theorem. □

In general, the proof of Theorem 3.2 is based on the fact that for any RE language L, there exist a homomorphism h and two linear languages L_1 and L_2 such that $L^R = h(L_1 \cap L_2)$. The bottom-up R-PDA recognizing L operates so that it first nondeterministically pushes some symbols onto its pushdown, and then verifies that its pushdown content, say γ, forms a string that belongs to L_1 and L_2, i.e., $\gamma \in L_1 \cap L_2$. After that verification, the automaton repeats reading a symbol, X, from the pushdown and $h(X)^R$ from the input.

Furthermore, it is known that the linear languages L_1 and L_2 can be of some special minimal forms, i.e., they belong to some proper subfamilies of the family of linear languages. For an overview of these minimal forms the reader is referred to Table 1 in [11].

In addition, the following result can be achieved by a simple modification of grammars G_1 and G_2 from Theorem 3.2 so that each production $S \to \alpha \in P_1$ of G_1 is replaced with $S \to \alpha c_i$, where c_i, for each $1 \leq i \leq |P_1|$, is a new symbol, and G_2 is modified in a corresponding way. Then it is not hard to see that the language $L(G_1)^R$ is linear and deterministic context-free, since for each production $S \to (c_i v_i S u_i)^R \in P_1$, the symbol c_i says that v_i^R be read from the input and u_i^R be pushed to the pushdown. Using these modified grammars and the notation $L_i = L(G_i)^R$, for $i = 1, 2$, we have the following corollary.

Corollary 3.2. Let L be an RE language. Then there are two linear and deterministic context-free languages L_1 and L_2, a regular language R, and a bottom-up $(L_1 \cup L_2 \cup R)$-PDA $\mathcal{M}$ such that $L = T(\mathcal{M})$.

It is an open problem whether there are such languages L_1, L_2, and R that the union $L_1 \cup L_2 \cup R$ is also a linear and deterministic context-free language. In other words, it is open whether any RE language can be recognized by a bottom-up R'-PDA, where R' is linear and deterministic context-free. This situation is different in the case of top-down R-PDAs.

Theorem 3.3. Let L be an RE language. Then there exist a linear and deterministic context-free control language R and a top-down R-PDA $\mathcal{M}$ such that $L = T(\mathcal{M})$.

Proof:
Let L_1 and L_2 be constructed as in the remark above of Corollary 3.2 (but with the modification that instead of $S \to \alpha c_i$ we have $S \to c_i\alpha$, for $S \to \alpha \in P$, the construction of L_2 is modified correspondingly, $L_i = L(G_i)$, $i = 1, 2$), i.e., we use the deterministic variants of languages defined in the proof of Theorem 3.2. Then we can see that $R = L_1 \cup L_2 \cup (\{A, B\} \cup T)^*$ is linear and deterministic context-free, since the strings of L_1 begins with two symbols \$, those of L_2 with only one \$, and $(\{A, B\} \cup T)^*$ is a regular language. □

Using the definitions and results of [11], we immediately obtain the following corollary. First, however, recall that a linear language $L \subseteq T^*$ is *minimal linear* if it is generated by a linear grammar $G = (N, T, P, S)$, where $N = \{S\}$ is a singleton set and G has a unique terminal production $S \to c$, where $c \in T$ appears only in this production. In addition, $G = (\{S\}, T, P, S)$ is $(1, 1)$-*minimal linear* if it is minimal linear and for each production $S \to \alpha S \beta \in P$, where $\alpha, \beta \in T^*$, $|\alpha| = |\beta| = 1$ is satisfied. A language is $(1, 1)$-minimal linear if it is generated by a $(1, 1)$-minimal linear grammar.

Corollary 3.3. Let L be an RE language. Then there exist a minimal linear language $L_1 \subseteq \Sigma^*$, a $(1, 1)$-minimal linear language $L_2 \subseteq \Sigma^*$, a regular language $R \subseteq \Sigma^*$, and a bottom-up $(L_1c_1 \cup L_2c_2 \cup R)$-PDA $\mathcal{M}$, where $c_1 \neq c_2$, $c_1, c_2 \notin \Sigma$, such that $L = T(\mathcal{M})$.

Proof:
It is proved in [11] that for every RE language L, there exist a minimal linear language $L_1 \subseteq \Sigma^*$, a $(1, 1)$-minimal linear language $L_2 \subseteq \Sigma^*$, and a homomorphism $h : \Sigma^* \to \Sigma^*$ such that $L^R = h(L_1 \cap L_2)$. Let $R = \Sigma^*$ be a regular language, $c_1, c_2 \notin \Sigma$ be two different symbols, and $(L_1c_1 \cup L_2c_2 \cup R)$-PDA $\mathcal{M}$ be constructed by the method discussed above. Here, c_i is used to check that the pushdown content forms a string belonging to L_i, for $i = 1, 2$, i.e., c_1 stands for \$\$ and c_2 for \$ in the proof of Theorem 3.2. Thus, $L = T(\mathcal{M})$ is satisfied. □

Corollary 3.4. Let L be an RE language. Then there exist a minimal linear language $L_1 \subseteq \Sigma^*$, a $(1, 1)$-minimal linear language $L_2 \subseteq \Sigma^*$, a regular language $R \subseteq \Sigma^*$, and a top-down $(c_1L_1 \cup c_2L_2 \cup R)$-PDA $\mathcal{M}$, where $c_1 \neq c_2$, $c_1, c_2 \notin \Sigma$, such that $L = T(\mathcal{M})$.

4. State-Controlled R-PDAs

From a practical point of view, it is obvious that the less checks of the pushdown content the automaton makes, the more efficient the computation can be. In R-PDAs, the pushdown content is checked in each computational step. However, taking a careful look at the proof of Theorem 3.2, we can see that only two checks are of interest: the first check is made when \$\$ is pushed onto the pushdown, and the other when the first \$ is removed. This observation motivates the following definition of R-sPDAs.

Let $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_c, Z_0, F)$ be a pushdown automaton, where $Q_c \subseteq Q$ is a set of *checking states*, and all other symbols are as in an ordinary pushdown automaton. Let $R \subseteq (\Gamma \setminus \{Z_0\})^*$ be a control language. Then $\mathcal{M}$ is called a bottom-up (top-down) *state-controlled* R-PDA (or R-sPDA for short) if for all $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

$$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma)$$

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

1. either $q \in Q \setminus Q_c$,
2. or $q \in Q_c$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \in R$ (or $\gamma'' \in R$ in the case of *top-down* R-sPDAs).

Note that if $q \in Q_c$ and $(\gamma'')^R \notin R$ (or $\gamma'' \notin R$, respectively), then there is no possible computational step and the automaton rejects the input.

The reader can imagine R-sPDAs as pushdown automata with an oracle which answers questions of whether the current content of the pushdown forms a string belonging to R.

The following theorem can be proved by the same technique used in the case of R-PDAs, where R is a regular control language, see Theorem 3.1.

Theorem 4.1. Let R be a regular language and $\mathcal{M}$ be a bottom-up (top-down) R-sPDA. Then an equivalent pushdown automaton $\mathcal{M}'$ can effectively be constructed.

Now, we can prove the following result concerning the case of non-regular control languages.

Theorem 4.2. Let L be an RE language. Then there exist a linear language R and a bottom-up (top-down) R-sPDA $\mathcal{M}$ such that $L = T(\mathcal{M})$. In addition, $\mathcal{M}$ checks the form of its pushdown content no more than twice during any computation.

Proof:
Consider the proof of Theorem 3.2 and modify the grammars G_1 and G_2 so that $L(G_1) = \{\$w_1w_2w : S \Rightarrow_G^* w_1' S w_2' w \Rightarrow_G w_1 w_2 w\}$ and $L(G_2) = \{w_1w_2w : w_1 \in \{AB, ABB\}^*, w_2 \in \{BBA, BA\}^*, w \in T^*\}$, where $w_1w_2w \in L(G_2)$ implies $w_1w_2w \Rightarrow^* w$ by the production $ABBBA \to \varepsilon$. Let $R = L(G_1)^R \cup L(G_2)^R$ be the linear control language (or $R = L(G_1) \cup L(G_2)$ in the case of top-down R-sPDAs), and define the R-sPDA $\mathcal{M} = (Q, T, \Gamma, \delta, q_0, \{q_c\}, Z_0, F)$ so that $Q = \{q_0, q_1, q_c, q_f\}$, $Q_c = \{q_c\}$, $\Gamma = T \cup \{A, B, \$, Z_0\}$, $F = \{q_f\}$, and δ is defined as follows: $\delta(q_0, \varepsilon, X) = \{(q_0, \alpha X) : \alpha \in T \cup \{A, B\}\}$, $\delta(q_0, \varepsilon, X) = \{(q_c, \$X)\}$, $\delta(q_c, \varepsilon, \$) = \{(q_c, \varepsilon)\}$, $\delta(q_c, \varepsilon, X) = \{(q_1, X)\}$, $\delta(q_1, \varepsilon, Y) = \{(q_1, \varepsilon)\}$, $\delta(q_1, a, a) = \{(q_1, \varepsilon)\}$, and $\delta(q_1, \varepsilon, Z_0) = \{(q_f, \varepsilon)\}$, where $X \in \{A, B, Z_0\} \cup T$, $Y \in \{A, B\}$, and $a \in T$. The proof now proceeds analogously to the proof of Theorem 3.2. □

As a corollary, we have the following descriptional complexity result.

Corollary 4.1. Let L be an RE language. Then there exist a linear language R and a bottom-up (top-down) R-sPDA $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_c, Z_0, F)$ which checks the form of the pushdown content no more than twice during any computation, such that $|Q| \leq 4$, $|Q_c| = 1$, $|\Gamma| \leq |\Sigma| + 4$, and $L = T(\mathcal{M})$.

The following result is similar to Corollary 3.2 (using the deterministic variants of those linear languages as discussed above of Corollary 3.2, where the construction of the proof of Theorem 4.2 is used instead of the construction of the proof of Theorem 3.2).

Corollary 4.2. Let L be an RE language. Then there exist two linear and deterministic context-free languages L_1, L_2, and a bottom-up $(L_1 \cup L_2)$-sPDA $\mathcal{M}$ which checks the form of the pushdown content no more than twice during any computation, such that $L = T(\mathcal{M})$.

Similarly as for bottom-up R-PDAs, it is an open problem whether any RE language can be recognized by a bottom-up R-sPDA, where R is linear and deterministic context-free.

On the other hand, for top-down R-sPDAs, we can see that using the combination of constructions of the proofs of Theorems 4.2 and 3.3 the language $L_1 \cup L_2$ is linear and deterministic context-free, since the strings of L_1 begins with \$ while those of L_2 does not. Thus, we have the following result.

Corollary 4.3. Let L be an RE language. Then there exist a linear and deterministic context-free language R and a top-down R-sPDA $\mathcal{M}$ which checks the form of the pushdown content no more than twice during any computation, such that $L = T(\mathcal{M})$.

Furthermore, using the results of [11], we have the following consequences.

Corollary 4.4. Let L be an RE language. Then there exist a minimal linear language $L_1 \subseteq \Sigma^*$, a $(1,1)$-minimal linear language $L_2 \subseteq \Sigma^*$, \$ $\notin \Sigma$, and a bottom-up $(L_1\$ \cup L_2)$-sPDA $\mathcal{M}$ which checks the form of the pushdown content no more than twice during any computation, such that $L = T(\mathcal{M})$.

Proof:
It is proved in [11] that for every RE language L, there exist a minimal linear language $L_1 \subseteq \Sigma^*$, a $(1,1)$-minimal linear language $L_2 \subseteq \Sigma^*$, and a homomorphism $h : \Sigma^* \to \Sigma^*$ such that $L^R = h(L_1 \cap L_2)$. Let \$ $\notin \Sigma$ be a new symbol, and let the bottom-up $(L_1\$ \cup L_2)$-sPDA $\mathcal{M}$ be constructed by the method discussed above of Corollary 3.2. Then $L = T(\mathcal{M})$. □

Corollary 4.5. Let L be an RE language. Then there exist a minimal linear language $L_1 \subseteq \Sigma^*$, a $(1,1)$-minimal linear language $L_2 \subseteq \Sigma^*$, \$ $\notin \Sigma$, and a top-down $(\$L_1 \cup L_2)$-sPDA $\mathcal{M}$ which checks the form of the pushdown content no more than twice during any computation, such that $L = T(\mathcal{M})$.

By a simple modification, Example 3.1 demonstrates that there is a bottom-up (top-down) R-sPDA, where R is linear and deterministic context-free, recognizing the non-context-free language $\{a^n b^n c^n d^n : n \geq 1\}$ with only one check of the pushdown content. However, the question of the computational power of R-sPDAs performing only one check is open.

4.1. Deterministic State-Controlled R-PDAs

In this section, we consider deterministic R-sPDAs (R-sDPDAs), which means that the core pushdown automaton of the R-sPDA is deterministic, with a linear (or linear and deterministic context-free) control language R.

As the core pushdown automata of R-sDPDAs are deterministic, these machines are able to recognize any deterministic context-free language. In addition, as these machines can work so that they only copy the whole input to the pushdown, it is obvious that any linear language L can be recognized by a bottom-up L-sDPDA (top-down L^R-sDPDA) $\mathcal{M}$ (assuming that the automaton can recognize the end of the input and go to the checking state). However, by a simple modification of Example 3.1, the following example illustrates that there are non-context-free languages that can be accepted by R-sDPDAs with a linear and deterministic context-free control language R and only one check of the pushdown content.

Example 4.1. Let $R = \{a^n b^{n-1} : n \geq 1\}$. Then R is linear and deterministic context-free. Let $\mathcal{M} = (\{q_a, q_b, q_c, q_d, q_{ch}, q_f\}, \{a, b, c, d\}, \{a, b, Z_0\}, \delta, q_a, \{q_{ch}\}, Z_0, \{q_f\})$ be a bottom-up R-sDPDA (top-down R^R-sDPDA) operating as follows:

1. Starting in q_a, $\mathcal{M}$ repeats reading a from the input and pushing a to the pushdown.
2. Reading the first b, $\mathcal{M}$ goes to state q_b and pushes b to the pushdown, i.e., the pushdown contains ba^nZ_0. Then, being in q_b, $\mathcal{M}$ deterministically repeats reading b from the input and pushing b to the pushdown.
3. Reading the first c, $\mathcal{M}$ goes to state q_{ch} and removes b from the pushdown top. In the next step, it checks that the pushdown content is of the form $b^{n-1}a^nZ_0$ and goes either to q_c, when reading c from the input and removing b from the pushdown, or to q_d, when reading d from the input and removing a from the pushdown.
4. Being in q_c, $\mathcal{M}$ repeats reading c from the input and removing b from the pushdown; being in q_c and having a on the top of the pushdown, $\mathcal{M}$ goes to q_d and repeats reading d from the input and removing a from the pushdown, i.e., c^n has been read.
5. Finally, being in q_d, $\mathcal{M}$ repeats reading d from the input and removing a from the pushdown; being in q_d and having Z_0 on the top of the pushdown, $\mathcal{M}$ goes to the final state q_f from which no other symbol can be read; moreover, nothing is read from the input and Z_0 is removed from the pushdown.

Thus, $T(\mathcal{M}) = \{a^nb^nc^nd^n : n \geq 1\}$, which is a non-context-free language.

Let L_1 and L_2 be two linear languages. Construct the following bottom-up $(L_1\$ \cup L_2)$-sDPDA (top-down $(\$L_1^R \cup L_2^R)$-sDPDA) $\mathcal{M}$ such that $T(\mathcal{M}) = L_1 \cap L_2$. $\mathcal{M}$ operates so that it first copies the whole input to the pushdown and then (assuming that $\mathcal{M}$ can recognize the end of the input and change its state) it goes to the checking state pushing \$ onto the top of the pushdown. Now, $\mathcal{M}$ checks that the pushdown content (the input, ignoring \$) belongs to L_1. If so, \$ is removed from the pushdown and $\mathcal{M}$ checks that the pushdown content also belongs to L_2. If both these checks are positive, we have that the input belongs to the intersection of those two linear languages. By a simple modification, we can generalize this method to a finite intersection of linear languages.

Corollary 4.6. Let $L_1, L_2, \ldots, L_n$ be linear languages, for some $n \geq 2$. Then there is a bottom-up $(L_1\$^{n-1} \cup L_2\$^{n-2} \cup \cdots \cup L_n)$-sDPDA (top-down $(\$^{n-1}L_1^R \cup \$^{n-2}L_2^R \cup \cdots \cup L_n^R)$-sDPDA) $\mathcal{M}$ recognizing the language $\bigcap_{i=1}^{n} L_i$.

The following theorem shows that the membership problem for R-sDPDAs with R being linear is decidable.

Theorem 4.3. If a language L is recognized by a bottom-up (top-down) R-sDPDA, for some linear control language R, then L is recursive.

Proof:
Let $\mathcal{M}$ be an R-PDA, and let $\mathcal{M}'$ be its core deterministic pushdown automaton. By the construction of Lemma 12.1 in [6], we can construct an equivalent deterministic pushdown automaton $\mathcal{M}''$ to $\mathcal{M}'$ which never performs an infinite loop. Note that $\mathcal{M}''$ has the same pushdown alphabet as $\mathcal{M}'$. Thus, replacing $\mathcal{M}'$ with $\mathcal{M}''$ in $\mathcal{M}$, we have an equivalent R-PDA to $\mathcal{M}$, denoted as $\mathcal{M}'''$. As the checks of the pushdown content can be performed by a Turing machine which always halts, since R is linear, and because $\mathcal{M}''$ always halts, we have that $\mathcal{M}'''$ always halts. □

Note that the computational power as well as all other properties of R-sDPDAs with a linear control language R are open. In addition, the power of these machines over a one-letter alphabet is an open problem, too. Are these machines powerful enough to recognize a non-regular language over a one-letter alphabet?

5. Conclusion

In this paper, we have shown that every RE language can be recognized by a bottom-up (top-down) R-PDA, where R is a non-regular, linear (and deterministic context-free, respectively) control language. In addition, only two checks of the form of the pushdown content are of interest during any computation of these machines. Based on this observation, a new type of R-PDAs has been introduced and discussed, so-called state-controlled R-PDAs. As an immediate consequence of the results concerning R-PDAs, we have that every RE language can be accepted by an R-sPDA which makes no more than two checks of the form of the pushdown content during any computation. On the other hand, it has also been shown that R-sPDAs with only one check of the pushdown content during any computation are powerful enough to recognize non-context-free languages. However, their precise computational power is left as an open problem.

Furthermore, from a practical point of view, it is of some interest to study the deterministic variant of R-sPDAs, where the core pushdown automaton is deterministic and the control language R is linear and deterministic context-free, since the languages recognized by those machines can be analyzed in linear time (assuming that the number of checks is bounded by a constant). It has been shown that there are non-context-free languages that can be accepted by R-sDPDAs with a linear and deterministic context-free control language R and with only one check of the pushdown content. It has also been proved that any deterministic context-free language and any language that can be written as a finite intersection of linear languages can be recognized by such a machine, and that any language recognized by such a machine is recursive. However, the precise computational power as well as all other properties are left open.

Finally, another interesting variant of these machines seems to be so-called visibly (also called input-driven) R-sPDAs, where the pushdown operations are driven by the input symbols (see [1, 2]). However, this is a part of the future research.

Acknowledgements

The author gratefully acknowledges useful suggestions and comments of the anonymous referees. This work was supported by the Czech Academy of Sciences, Institutional Research Plan no. AV0Z10190503.

References

[1] Alur, R., Madhusudan, P.: Visibly pushdown languages, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing* (L. Babai, Ed.), ACM, 2004.

[2] Bollig, B.: On the expressive power of 2-stack visibly pushdown automata, *Logical Methods in Computer Science*, **4**(4), 2008, 1–35.

[3] Dassow, J., Păun, G.: *Regulated Rewriting in Formal Language Theory*, Springer, Berlin, 1989.

[4] Dassow, J., Păun, G., Salomaa, A.: Grammars with controlled derivations, *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, Eds.), 2, Springer, Berlin, 1997.

[5] Geffert, V.: Normal Forms for Phrase-Structure Grammars, *RAIRO – Theoretical Informatics and Applications*, **25**(5), 1991, 473–496.

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

[7] Kolář, D., Meduna, A.: Regulated Pushdown Automata, *Acta Cybernetica*, **4**, 2000, 653–664.

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

[9] Kutrib, M., Malcher, A., Werlein, L.: Regulated Nondeterminism in Pushdown Automata, *Theoretical Computer Science*, **410**(37), 2009, 3447–3460.

[10] Meduna, A., Kolář, D.: One-Turn Regulated Pushdown Automata and Their Reduction, *Fundamenta Informaticae*, **51**(4), 2002, 399–405.

[11] Okawa, S., Hirose, S.: Homomorphic characterizations of recursively enumerable languages with very small language classes, *Theoretical Computer Science*, **250**(1-2), 2001, 55–69.

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

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