Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

A PUMPING LEMMA FOR FLIP-PUSHDOWN LANGUAGES *

Peter Kostolányi[1]

Abstract. Flip-pushdown automata are pushdown automata with an extra ability to reverse the contents of the pushdown store. A generalisation of the Pumping lemma for context-free languages is presented, which applies to the families of languages accepted by flip-pushdown automata with k pushdown flips, for an arbitrary constant k. The presented result gives rise to a new technique for disproving existence of flip-pushdown automata with a constant number of flips, which is significantly simpler compared to methods used for this purpose so far.

1991 Mathematics Subject Classification. 68Q45.

Introduction

Flip-pushdown automata can be described as ordinary nondeterministic pushdown automata with special transitions that can be used to reverse – or *flip* – the contents of the pushdown store. The model as such has been introduced by Sarkar [16], and many of its fundamental properties have been resolved soon after by Holzer and Kutrib [11, 12].

Already Sarkar has observed [16] that flip-pushdown automata with an unbounded number of pushdown flips are Turing-complete. For this reason, the research has focused mainly on the setting, in which the number of flips is viewed as a limited computational resource. In particular, the most interesting results so far have been obtained on flip-pushdown automata with the number of pushdown flips limited by a constant. By the Hierarchy theorem of Holzer and Kutrib [11],

Keywords and phrases: Pumping lemma; Flip-pushdown automaton; Flip-pushdown language; Reversal-generating context-free grammar

* *This work has been supported in part by the grant VEGA 1/0967/16.*

[1] Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia

flip-pushdown automata with $k+1$ flips are (strictly) stronger than flip-pushdown automata with k flips.

With the aim to provide some new proof techniques, two families of grammars have recently been introduced by the present author [14], which are equivalent to flip-pushdown automata with a constant number of flips: *reversal-generating context-free grammars* and *parallel interleaving grammar systems*. The latter grammatical characterisation has already been applied in [14] to resolve the relation between flip-pushdown automata with a constant number of flips and ET0L-systems, answering a question of Holzer and Kutrib [11].

In this paper, we take advantage of the characterisation in terms of reversal-generating grammars and prove a pumping lemma for languages accepted by flip-pushdown automata with k flips, for all $k \geq 1$. This pumping lemma can be viewed as a generalisation of the classical Pumping lemma for context-free languages [2, 13], and we demonstrate that it may be used to disprove the existence of flip-pushdown automata for some specific languages. Such proofs appear to be significantly simpler compared to proofs via previously known techniques. We provide some representative examples in a separate section.

In addition, we show that the presented Pumping lemma for flip-pushdown languages is optimal, in the sense that the bounds on subword lengths occurring in its statement cannot be improved any further.

1. Flip-Pushdown Automata

Let us briefly review the fundamental definitions and results related to flip-pushdown automata, which we shall use later in this paper. The following definition appeared for the first time in the paper of Sarkar [16].

Definition 1.1. A *(nondeterministic) flip-pushdown automaton* (abbr. NFPDA) is an octuple $A = (K, \Sigma, \Gamma, \delta, \Delta, q_0, Z_0, F)$, where K is a finite set of states, Σ is an input alphabet, Γ is a pushdown alphabet, δ is an "ordinary" transition function from $K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subsets of $K \times \Gamma^*$, Δ is a flip transition function from K to subsets of K, q_0 in K is an initial state, Z_0 in Γ is a bottom-of-pushdown symbol, and $F \subseteq K$ is a set of accepting states.

A *configuration* of the NFPDA A is a triple (q, w, s) in $K \times \Sigma^* \times \Gamma^*$, interpreted in the same way as for "ordinary" PDA.

A *computation step* of the NFPDA A is a relation $\vdash_A$ on its configurations, defined separately for "ordinary" transitions (in the same way as for PDA), and for flip transitions, which result in flipping the pushdown store. The formal definition goes as follows: let p, q be in K, a be in $\Sigma \cup \{\varepsilon\}$, u be in Σ^*, s, t be in Γ^*, and Z be in Γ. For *"ordinary" transitions*, we define $(p, au, sZ) \vdash_A (q, u, st)$ if (q, t) is in $\delta(p, a, Z)$. For *flip transitions*, we define $(p, u, Z_0 s) \vdash_A (q, u, Z_0 s^R)$ if q is in $\Delta(p)$. If A is understood, we write $\vdash$ instead of $\vdash_A$.

Remark 1.2. Two details in the definition of the computation step call for special attention. First, when a flip transition is executed, the pushdown store is

reversed *except for the bottom-of-pushdown symbol*. However, this (usual) definition is clearly equivalent to an alternative one, in which the entire pushdown store is reversed. Indeed, let us abbreviate such "alternative" flip-pushdown automata by NFPDA$'$. Then it is clear that each NFPDA$'$ can be simulated by an NFPDA – it suffices to add a new bottom-of-pushdown symbol, which is not altered during the computation. Conversely, each NFPDA can be simulated by an NFPDA$'$ – instead of a flip transition of the NFPDA (which does not change the position of Z_0 on the bottom of the pushdown), it is sufficient to push Z_0 to the top of the pushdown, then flip the pushdown store (including the bottom-of-pushdown symbol), and finally pop Z_0 from its top. It is clear that in both simulations above, the number of pushdown flips executed remains unchanged.

Secondly, flip transitions can be executed only when Z_0 takes place on the bottom of the pushdown store. Once again, this is the usual definition, which poses no significant restriction: each flip-pushdown automaton can be transformed into a normal form, in which the symbol on the bottom of the pushdown is *always* Z_0.

The *language* $L(A)$ accepted by A *by final state* is defined, similarly as for "ordinary" PDA, by

$$L(A) = \{w \in \Sigma^* \mid \exists q \in F, s \in \Gamma^* : (q_0, w, Z_0) \vdash^* (q, \varepsilon, s)\},$$

and the *language* $N(A)$ accepted *by empty pushdown* is defined by

$$N(A) = \{w \in \Sigma^* \mid \exists q \in K : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}.$$

We say that the NFPDA A *operates in at most (exactly) k flips*, if in every of its computations, flip transitions are performed at most (exactly) k times.

Several results relating families of NFPDAs are known up to now. Holzer and Kutrib have proved [11] the equivalence of NFPDA accepting by final state and NFPDA accepting by empty pushdown, and have argued that the simulations involved do not change the number of flips performed. Furthermore, NFPDA operating in at most k flips are proved to be equivalent to NFPDA operating in exactly k flips [11]. Finally, without any restriction on the number of pushdown flips, NFPDA are known to be equivalent to Turing machines [16].

Definition 1.3. The family of languages accepted by NFPDA operating in k flips is denoted by $\mathscr{L}(\text{NFPDA}_k)$. Furthermore,

$$\mathscr{L}(\text{NFPDA}_{fin}) := \bigcup_{k=0}^{\infty} \mathscr{L}(\text{NFPDA}_k).$$

The family of languages accepted by arbitrary NFPDA, equal to the family of recursively enumerable languages [16], may occasionally be denoted by $\mathscr{L}(\text{NFPDA})$.

Remark 1.4. This definition of the families $\mathscr{L}(\text{NFPDA}_k)$ slightly differs from the original definition used, e.g., by Holzer and Kutrib [11,12]. There, a language $L_k(A)$ is defined for every k and *for every* flip-pushdown automaton A – it consists

of all words accepted by A in at most k pushdown flips. The family $\mathscr{L}(\mathrm{NFPDA}_k)$ is then defined to contain languages $L_k(A)$ for all flip-pushdown automata A (and to contain no other languages). However, it is clear that for all NFPDA A operating in k pushdown flips we have $L_k(A) = L(A)$. Conversely, to any NFPDA A it is possible to construct an NFPDA A' operating in k pushdown flips, such that $L(A') = L_k(A)$ – this may store the number of pushdown flips executed so far in its state and reject if this number should exceed k.

This means that our definition of the families $\mathscr{L}(\mathrm{NFPDA}_k)$ is equivalent to the original definition used previously.

Holzer and Kutrib have proved [11] that the families $\mathscr{L}(\mathrm{NFPDA}_k)$ form an infinite hierarchy with respect to k. We shall refer to this fundamental result as to the *Hierarchy theorem*.

Theorem 1.5 (Holzer and Kutrib [11])**.** *The families $\mathscr{L}(\mathrm{NFPDA}_k)$ form an infinite hierarchy with respect to k:*

$$\mathscr{L}(CF) = \mathscr{L}(\mathrm{NFPDA}_0) \subsetneq \mathscr{L}(\mathrm{NFPDA}_1) \subsetneq \mathscr{L}(\mathrm{NFPDA}_2) \subsetneq \dots$$

Finally in this section, let us state the important *Flip-pushdown input-reversal theorem*, proved by Holzer and Kutrib in [11] (the theorem has been originally stated in a slightly different form).

Theorem 1.6 (Holzer and Kutrib [11])**.** *Let k be in $\mathbb{N}$. A language L is accepted by empty pushdown by a NFPDA $A_1 = (K, \Sigma, \Gamma, \delta, \Delta, q_0, Z_0, \emptyset)$ operating in $k+1$ pushdown flips if and only if the language*

$$L_R = \{uv^R \mid (q_0, u, Z_0) \vdash_{A_1}^* (q_1, \varepsilon, Z_0 s) \text{ with } k \text{ flips, } q_2 \in \Delta(q_1), \\ \text{and } (q_2, v, Z_0 s^R) \vdash_{A_1}^* (q_3, \varepsilon, \varepsilon) \text{ without any flip}\}$$

is accepted by empty pushdown by some NFPDA A_2 operating in k pushdown flips. The same statement holds for NFPDA accepting by final state.

2. Reversal-Generating Grammars

Two characterisations of flip-pushdown automata in terms of grammars have recently been introduced by the present author [14]. In this paper, we shall make use of the characterisation relating NFPDA to reversal-generating context-free grammars (RGCFG). Essentially, RGCFGs can be described as "ordinary" context-free grammars with an extra ability to generate reversals along with ordinary terminal symbols. In what follows, we shall briefly review some of the basic definitions and results related to RGCFGs.

Definition 2.1. A *reversal-generating context-free grammar* (abbr. RGCFG) is a quintuple $G = (N, T, P, \sigma, ®)$, where (N, T, P, σ) is a context-free grammar, and ® in T is a special reversal symbol.

The definition of a *derivation step* is the same as for context-free grammars – that is, for a reversal-generating grammar $G = (N, T, P, \sigma, ®)$, we write $u \Rightarrow_G v$ if and only if $u \Rightarrow_{G'} v$, where G' is the context-free grammar $G' = (N, T, P, \sigma)$.

For a RGCFG $G = (N, T, P, \sigma, ®)$, we shall denote by $L_{CF}(G)$ the language $L(G')$ generated by the context-free grammar $G' = (N, T, P, \sigma)$. The *language* generated by the RGCFG $G = (N, T, P, \sigma, ®)$ consists of words generated by the context-free grammar G' – that is of words from $L_{CF}(G)$ – with the ®-symbols interpreted in the left-to-right order as reversals of the remaining part of the word. Formally,

$$L(G) = \{\varrho(w) \mid w \in L_{CF}(G)\},$$

where $\varrho : T^* \to (T - \{®\})^*$ is the reversal-interpreting function defined by

$$\varrho(w) = \begin{cases} w & \text{for } w \text{ without an occurrence of } ®, \\ u\varrho(v^R) & \text{for } w = u®v,\ u \text{ without an occurrence of } ®, \text{ and } v \text{ in } T^*. \end{cases}$$

For c in T and w in T^*, we denote by $|w|_c$ the number of occurrences of c in w. The RGCFG G is said to *generate at most (exactly) k reversals*, if for all words w from $L_{CF}(G)$, $|w|_® \leq k$ ($|w|_® = k$).

Let us now establish some simple properties of reversal-generating context-free grammars. First, let us prove a proposition relating the language $L(G)$ to the language $L_{CF}(G)$.

Proposition 2.2. *Let $G = (N, T, P, \sigma, ®)$ be a* RGCFG*. Then*

$$\begin{aligned} L(G) = &\{w_1 w_{2n}^R w_2 w_{2n-1}^R \dots w_n w_{n+1}^R \mid w_1 ® w_2 ® \dots ® w_{2n} \in L_{CF}(G)\} \cup \\ &\cup \{w_1 w_{2n+1}^R w_2 w_{2n}^R \dots w_n w_{n+2}^R w_{n+1} \mid w_1 ® w_2 ® \dots ® w_{2n+1} \in L_{CF}(G)\}. \end{aligned}$$

Proof. By definition, $L(G) = \{\varrho(w) \mid w \in L_{CF}(G)\}$, where ϱ is the reversal-interpreting function. Thus, it is sufficient to prove that for all n in $\mathbb{N}$ and all words $w_1, \dots, w_{2n+1}$ in $(T - \{®\})^*$,

$$\varrho(w_1 ® w_2 ® \dots ® w_{2n}) = w_1 w_{2n}^R w_2 w_{2n-1}^R \dots w_n w_{n+1}^R$$

and

$$\varrho(w_1 ® w_2 ® \dots ® w_{2n+1}) = w_1 w_{2n+1}^R w_2 w_{2n}^R \dots w_n w_{n+2}^R w_{n+1}.$$

By induction on n. The claim is trivial for $n = 0$, as $\varrho(\varepsilon) = \varepsilon$ and $\varrho(w_1) = w_1$.

Now, let us suppose that the claim holds for $n = k$. We shall prove that it holds for $n = k + 1$. Indeed, we have

$$\begin{aligned} \varrho(w_1 ® w_2 ® \dots ® w_{2k+2}) &= w_1 \varrho(w_{2k+2}^R ® w_{2k+1}^R ® \dots ® w_2^R) = \\ &= w_1 w_{2k+2}^R w_2 w_{2k+1}^R \dots w_{k+1} w_{k+2}^R, \end{aligned}$$

where the first equality is by definition of the reversal-interpreting function ϱ and the second equality follows by the induction hypothesis. Similarly,

$$\varrho(w_1 ® w_2 ® \dots ® w_{2k+3}) = w_1 \varrho(w_{2k+3}^R ® w_{2k+2}^R ® \dots ® w_2^R) = \\ = w_1 w_{2k+3}^R w_2 w_{2k+2}^R \dots w_{k+3}^R w_{k+2},$$

where the first equality is by the definition of ϱ and the second equality follows by what we have proved above. The proposition is proved. □

The following proposition implies that the "usual" normal forms for context-free grammars, such as the Chomsky normal form [5, 13], generalize directly to reversal-generating grammars.

Proposition 2.3. *Let $G_1 = (N_1, T_1, P_1, \sigma_1, ®_1)$ and $G_2 = (N_2, T_2, P_2, \sigma_2, ®_2)$ be* RGCFG*s. If $L_{CF}(G_1) = L_{CF}(G_2)$, then $L(G_1) = L(G_2)$.*

Proof. Let $L_{CF}(G_1) = L_{CF}(G_2)$. Then it follows that

$$L(G_1) = \{\varrho(w) \mid w \in L_{CF}(G_1)\} = \{\varrho(w) \mid w \in L_{CF}(G_2)\} = L(G_2)$$

and the proposition is proved. □

Each reversal-generating context-free grammar $G = (N, T, P, \sigma, ®)$ unambiguously determines an "ordinary" context-free grammar $G' = (N, T, P, \sigma)$, such that $L(G') = L_{CF}(G)$. To transform the "ordinary" context-free grammar G' into some normal form means to construct an "ordinary" context-free grammar G'', such that $L(G'') = L(G') = L_{CF}(G)$ and G'' satisfies some condition. However, we can view the grammar G'' as a reversal-generating context-free grammar G''', for which Proposition 2.3 implies $L(G''') = L(G)$.

As a result, every reversal-generating context-free grammar can be transformed, e.g., into the Chomsky normal form [5,13], into the Greibach normal form [10,13], or into the Double Greibach normal form [9,15]. Moreover, the number of reversal symbols generated by the grammars obviously remains unchanged under these transformations.

Remark 2.4. The converse of the implication of Proposition 2.3 does not hold. To see this, let us consider a reversal-generating grammar $G_1 = (N_1, T_1, P_1, \sigma_1, ®_1)$, such that $N_1 = \{\sigma_1\}$, $T_1 = \{a, ®\}$, and $P_1 = \{\sigma_1 \to a\}$. Moreover, let us take a reversal-generating grammar $G_2 = (N_2, T_2, P_2, \sigma_2, ®_2)$, such that $N_2 = \{\sigma_2\}$, $T_2 = \{a, ®\}$, and $P_2 = \{\sigma_2 \to a®\}$. As can be easily observed, we obtain $L_{CF}(G_1) = \{a\} \neq \{a®\} = L_{CF}(G_2)$. However, $L(G_1) = L(G_2) = \{a\}$.

Next, we shall prove that for each RGCFG generating *at most* k reversals, there is an equivalent RGCFG generating *exactly* k reversals. So the situation appears to be similar as for flip-pushdown automata, where NFPDA operating in exactly k pushdown flips have the same computational power as NFPDA operating in at most k pushdown flips.

Proposition 2.5. *Let G_1 be a* RGCFG *generating* at most k *reversal symbols. Then a* RGCFG G_2 *exists, such that $L(G_2) = L(G_1)$ and G_2 generates* exactly k *reversal symbols.*

Proof. Let $G = (N, T, P, \sigma, ®)$ be a RGCFG in Chomsky normal form, generating at most k reversals, such that $L(G) = L(G_1)$. We shall construct a RGCFG $G_2 = (N_2, T, P_2, \sigma_2, ®)$, generating exactly k reversals, such that $L(G_2) = L(G)$.

The main idea is that in each derivation, the grammar G_2 first "guesses" the number $m \leq k$ of reversal symbols to be generated by the grammar G. Next, the derivation proceeds in the same way as in the grammar G, but it is allowed to generate a terminal word only if it indeed produces m reversal symbols – that is, the "guess" from the beginning has to be verified.

At the end of the derivation, $k-m$ additional reversal symbols are produced at the position where they have no effect on the interpretation of the generated word. By Proposition 2.2, this is always alongside the $\lceil (m+1)/2 \rceil$-th reversal symbol of the original generated word.

In order to add the right number of reversal symbols in the end, the number m has to be stored as an additional component in nonterminals used in the derivation after making the initial "guess".

Moreover, in order to be able to identify the $\lceil (m+1)/2 \rceil$-th reversal symbol, the nonterminals are further extended by numbers $1 \leq i \leq j \leq m$, meaning that the given nonterminal generates the i-th to the j-th reversal symbol of the resulting word. If a nonterminal generates no reversal symbols, then $i = j = 0$. When a nonterminal is being rewritten to two nonterminals, it is "guessed" how these values are distributed between them. These "guesses" are then verified when producing reversal symbols (then $i = j \neq 0$ has to hold) and "ordinary" terminal symbols (then $i = j = 0$ has to hold). This also verifies the number m "guessed" in the beginning of the derivation.

Formally, $N_2 = \{\sigma_2\} \cup N \times \{1, \ldots, k\}^3 \cup N \times \{1, \ldots, k\} \times \{(0,0)\}$, σ_2 is a new nonterminal, and the set of production rules P_2 is given by

$$
\begin{aligned}
P_2 = {} & \{\sigma_2 \to (\sigma, 0, 0, 0)®^k\} \cup \{\sigma_2 \to (\sigma, m, 1, m) \mid 1 \leq m \leq k\} \\
\cup {} & \{(\xi, m, i, j) \to (\alpha, m, i, t)(\beta, m, t+1, j) \mid 1 \leq i \leq t < j \leq m \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, m, i, j) \to (\alpha, m, i, j)(\beta, m, 0, 0) \mid 1 \leq i \leq j \leq m \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, m, i, j) \to (\alpha, m, 0, 0)(\beta, m, i, j) \mid 1 \leq i \leq j \leq m \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, m, 0, 0) \to (\alpha, m, 0, 0)(\beta, m, 0, 0) \mid 0 \leq m \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, m, \lceil (m+1)/2 \rceil, \lceil (m+1)/2 \rceil) \to ®^{k-m+1} \mid 1 \leq m \leq k;\ \xi \to ® \in P\} \\
\cup {} & \{(\xi, m, i, i) \to ® \mid 1 \leq i \leq m \leq k;\ i \neq \lceil (m+1)/2 \rceil;\ \xi \to ® \in P\} \\
\cup {} & \{(\xi, m, 0, 0) \to c \mid 0 \leq m \leq k;\ c \in (T - \{®\}) \cup \{\varepsilon\};\ \xi \to c \in P\}.
\end{aligned}
$$

It should be clear that $L(G_2) = L(G)$ and that G_2 always generates exactly k reversal symbols. The proposition is proved. □

We shall call the normal form introduced by the following proposition the *Reversal-aware normal form*. Intuitively, a reversal-generating grammar is in the reversal-aware normal form if each nonterminal always generates the same occurrences of ®-symbols. That is, a nonterminal is always "aware" of which reversal symbols it produces.

Proposition 2.6. *Let $G_1 = (N_1, T, P_1, \sigma_1, ®)$ be a* RGCFG *generating exactly k reversals. Then a* RGCFG $G_2 = (N_2, T, P_2, \sigma_2, ®)$ *generating k reversals exists, such that $L(G_2) = L(G_1)$ and for each ξ in N_2, one of the following two properties holds:*

- *(i) The nonterminal ξ does not generate reversal symbols. That is, $|w|_® = 0$ holds for all w in T^*, such that $\xi \Rightarrow^* w$.*
- *(ii) Numbers i, j in $\{1, \ldots, k\}$ do exist, such that the nonterminal ξ always generates the i-th to the j-th reversal symbol of the final word. That is, for all x, y, w in T^*, such that $\sigma_2 \Rightarrow^* x\xi y \Rightarrow^* xwy$, $|x|_® = i - 1$, and $|y|_® = k - j$.*

Proof. The idea is very similar to the one used in the proof of Proposition 2.5 – to each nonterminal, we add two numbers i and j providing information about the reversal symbols it produces. Such a grammar is obviously in the reversal-aware normal form.

Formally, let $G = (N, T, P, \sigma, ®)$ be a RGCFG in Chomsky normal form, equivalent to G_1 and generating exactly k reversals. We shall construct the grammar $G_2 = (N_2, T, P_2, \sigma_2, ®)$ as follows: $N_2 = N \times \{1, \ldots, k\}^2 \cup N \times \{(0,0)\}$, $\sigma_2 = (\sigma, 1, k)$, and

$$\begin{aligned}
P_2 = {} & \{(\xi, i, j) \to (\alpha, i, t)(\beta, t+1, j) \mid 1 \leq i \leq t < j \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, i, j) \to (\alpha, i, j)(\beta, 0, 0) \mid 1 \leq i \leq j \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, i, j) \to (\alpha, 0, 0)(\beta, i, j) \mid 1 \leq i \leq j \leq k;\ \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, 0, 0) \to (\alpha, 0, 0)(\beta, 0, 0) \mid \xi \to \alpha\beta \in P\} \\
\cup {} & \{(\xi, i, i) \to ® \mid 1 \leq i \leq k;\ \xi \to ® \in P\} \\
\cup {} & \{(\xi, 0, 0) \to c \mid c \in (T - \{®\}) \cup \{\varepsilon\};\ \xi \to c \in P\}.
\end{aligned}$$

The grammar G_2 is obviously equivalent to G and in the reversal-aware normal form. The proposition is proved. □

The families of languages generated by reversal-generating context-free grammars are denoted in analogy with the families of languages accepted by flip-pushdown automata.

Definition 2.7. The family of languages generated by RGCFG producing k reversal symbols is denoted by $\mathscr{L}(\mathrm{RGCFG}_k)$. Furthermore,

$$\mathscr{L}(\mathrm{RGCFG}_{fin}) := \bigcup_{k=0}^{\infty} \mathscr{L}(\mathrm{RGCFG}_k).$$

The following theorem proved by the present author [14] asserts that RGCFG producing k reversal symbols are equivalent to flip-pushdown automata operating in k flips.

Theorem 2.8 (K. [14])**.** *For all k in $\mathbb{N}$, $\mathscr{L}(\mathrm{NFPDA}_k) = \mathscr{L}(\mathrm{RGCFG}_k)$ holds. As a direct consequence, $\mathscr{L}(\mathrm{NFPDA}_{fin}) = \mathscr{L}(\mathrm{RGCFG}_{fin})$.*

Proof. Follows by Lemma 2.9, by Lemma 2.10, and by the fact that NFPDA accepting by empty pushdown store have the same computational power as NFPDA accepting by final state [11]. □

Lemma 2.9. *Let k be in $\mathbb{N}$ and $G = (N, T, P, \sigma, ®)$ be a* RGCFG *generating at most k reversals. Then a flip-pushdown automaton $A = (K, \Sigma, \Gamma, \delta, \Delta, q_0, Z_0, F)$ exists, such that A operates in at most k pushdown flips and $N(A) = L(G)$.*

Proof. We shall modify the standard simulation [13] of a leftmost derivation of an "ordinary" context-free grammar by an "ordinary" pushdown automaton. During the entire computation of A, a special symbol Z_0 will be placed on the bottom of the pushdown store – the simulation itself will take place above this special symbol. If an "ordinary" terminal symbol c in $T - \{®\}$ is found at the top of the pushdown, the automaton A pops this symbol from the pushdown and reads the same symbol from the input. If ® is found at the top of the pushdown, the automaton A flips its pushdown store. Finally, if a nonterminal ξ in N is found at the top of the pushdown, the automaton A nondeterministically rewrites it according to some production rule $\xi \to x \in P$ – if the number of pushdown flips performed so far is even, then ξ is replaced by x and if the number of flips is odd, then ξ is replaced to x^R. This means that the parity of the number of pushdown flips performed (or the number of ®-symbols processed) has to be stored in the state of the automaton A.

Formally, let us take $K = \{q_{init}, q_{even}, q_{odd}, q'_{even}, q'_{odd}, q_{fin}\}$, $\Sigma = T - \{®\}$, $\Gamma = N \cup T \cup \{Z_0\}$, $q_0 = q_{init}$, let Z_0 be a new symbol, $F = \emptyset$, and let the transition functions δ and Δ be given as follows:

$$
\begin{aligned}
\delta(q_{init}, \varepsilon, Z_0) &\ni (q_{even}, Z_0\sigma), \\
\delta(q_{even}, c, c) &\ni (q_{even}, \varepsilon) && \text{for all } c \text{ in } T - \{®\}, \\
\delta(q_{odd}, c, c) &\ni (q_{odd}, \varepsilon) && \text{for all } c \text{ in } T - \{®\}, \\
\delta(q_{even}, \varepsilon, ®) &\ni (q'_{even}, \varepsilon), \\
\delta(q_{odd}, \varepsilon, ®) &\ni (q'_{odd}, \varepsilon), \\
\Delta(q'_{even}) &\ni \{q_{odd}\}, \\
\Delta(q'_{odd}) &\ni \{q_{even}\}, \\
\delta(q_{even}, \varepsilon, \xi) &\ni (q_{even}, x^R) && \text{for all } \xi \to x \text{ in } P, \\
\delta(q_{odd}, \varepsilon, \xi) &\ni (q_{odd}, x) && \text{for all } \xi \to x \text{ in } P, \\
\delta(q_{even}, \varepsilon, Z_0) &\ni (q_{fin}, \varepsilon), \\
\delta(q_{odd}, \varepsilon, Z_0) &\ni (q_{fin}, \varepsilon).
\end{aligned}
$$

No further transitions are defined in A. Let us note that the same automaton with $F = \{q_{fin}\}$ instead of $F = \emptyset$ accepts the same language *by accepting state*.

Without loss of generality, we may suppose that the original grammar G is in the Reduced normal form (see, e.g., [13]), in which at least one terminal word can be derived from each sentential form (to be more precise, this is not true if $L(G) = \emptyset$, but this case is trivial). Then, obviously, each sentential form of G contains at most k reversal symbols. As a result, A operates in at most k pushdown flips. □

Lemma 2.10. *Let k be in $\mathbb{N}$ and $A = (K, \Sigma, \Gamma, \delta, \Delta, q_0, Z_0, F)$ be an* NFPDA *operating in exactly k pushdown flips. Then a reversal-generating context-free grammar $G = (N, T, P, \sigma, ®)$ exists, such that G generates exactly k reversals and $L(G) = L(A)$*

Proof. By a minor change to the transition function of A, it is obviously possible to obtain a flip-pushdown automaton A', which operates in the same way as A, but which has to read some new special symbol # before each pushdown flip (and the only thing A' may do after reading # is to flip the pushdown store).

Formally, the construction goes as follows: $A' = (K', \Sigma', \Gamma', \delta', \Delta', q_0', Z_0', F')$, where $K' = K \times \{1, 2\}$, $\Sigma' = \Sigma \cup \{\#\}$, $\Gamma' = \Gamma$, $q_0' = (q_0, 1)$, $Z_0' = Z_0$, $F' = F \times \{1\}$, and the transition functions δ' and Δ' are given as follows:

$$\begin{aligned}
\delta'((q,1),c,Z) &= \{((p,1),t) \mid (p,t) \in \delta(q,c,Z)\} && \text{for all } q \text{ in } K, c \text{ in } \Sigma, Z \text{ in } \Gamma,\\
\delta'((q,1),\#,Z) &= \{((p,2),Z) \mid p \in \Delta(q)\} && \text{for all } q \text{ in } K, Z \text{ in } \Gamma,\\
\Delta'((q,2)) &= \{(q,1)\} && \text{for all } q \text{ in } K.
\end{aligned}$$

The language $L(A')$ obviously consists of words $u = u_1\#u_2\#\ldots\#u_{k+1}$, such that $u_1u_2\ldots u_{k+1}$ is accepted by A with k pushdown flips on the positions marked by #. If k is odd, it is easy to prove by induction that by applying Theorem 1.6 k times, we obtain the language

$$L' = \{u_1\#u_3\#\ldots\#u_k\#u_{k+1}^R\#u_{k-1}^R\#\ldots\#u_2^R \mid u_1\#u_2\#\ldots\#u_{k+1} \in L(A')\}.$$

Similarly, if k is even, the k-fold application of Theorem 1.6 yields the language

$$L' = \{u_1\#u_3\#\ldots\#u_{k+1}\#u_k^R\#u_{k-2}^R\#\ldots\#u_2^R \mid u_1\#u_2\#\ldots\#u_{k+1} \in L(A')\}.$$

By Theorem 1.6, the language L' is context-free in both cases. Let us now consider a context-free grammar G', such that $L(G') = L'$. Let G be a reversal-generating context-free grammar defined in the same way as G', except that it produces reversal symbols ® instead of symbols #. It then follows by Proposition 2.2 that

$$L(G) = \{u_1u_2\ldots u_{k+1} \mid u_1\#u_2\#\ldots\#u_{k+1} \in L(A')\} = L(A),$$

and the lemma is proved. □

3. The Pumping Lemma

Let us now present the main result of this paper: the Pumping lemma for flip-pushdown languages. Our pumping lemma is a generalisation of the well-known Pumping lemma for context-free languages [2] (for an expository treatment, see, e.g., [13]), and can be used as an efficient tool for proving that a given language is not in $\mathscr{L}(\mathrm{NFPDA}_k)$ for some particular k (or for all k). In the proof of the Pumping lemma, we shall rely on Theorem 2.8 – we shall base our argumentation on reversal-generating grammars instead of directly on flip-pushdown automata.

The classical Pumping lemma for context-free languages can be stated either with two constants p and q, or with only one constant, which can be chosen to be the maximum of p and q. The situation in our case is similar. For aesthetic reasons, we shall state the Pumping lemma for flip-pushdown languages with only one constant q, but it can be restated with three different constants as well.

Theorem 3.1. *Let $k \geq 1$ be a positive integer and L be a language in $\mathscr{L}(\mathrm{NFPDA}_k)$. Then a nonnegative integer q exists, such that for all w in L, $|w| \geq q$, words x, u, y, v, z can be found, satisfying the following conditions:*

1. *The word w can be factored as $w = xuyvz$.*
2. *Either $|uyv| \leq q$, or $|uyv| \geq \frac{|w|}{2k} - q$ with $|uv| \leq q$.*
3. *The word uv is nonempty.*
4. *For all i in $\mathbb{N}$, xu^iyv^iz is in L.*

Proof. Let L be in $\mathscr{L}(\mathrm{NFPDA}_k)$. We shall specify q at the end of the proof, although we shall make several assumptions of the form $q \geq q'$ during the course of the proof. By Theorem 2.8, Proposition 2.5, and Proposition 2.6, there is a reversal-generating context-free grammar $G' = (N', T, P', \sigma, ®)$ in the Reversal-aware normal form and always generating exactly k reversals, such that $L(G') = L$. The Reversal-aware normal form allows us to divide the set of nonterminals of G' into *reversal-generating* and *non-reversal-generating nonterminals* – we shall rely on this distinction in what follows.

Let w be a word in L and w' be a word in $L_{CF}(G')$, such that one gets w after interpreting the reversal symbols in w'. Let a derivation tree R' of w' in G' be fixed. Let q_1 be a constant that is guaranteed for G', viewed as a context-free grammar, by the classical Pumping lemma for context-free languages.[1] We shall later choose the value of q so that $q \geq q_1$.

Now, suppose there is a node in the derivation tree R', corresponding to some *non-reversal-generating* nonterminal ξ, such that at least q terminal symbols are among its descendants – that is $\sigma \Rightarrow^* r\xi t \Rightarrow^* rst = w'$ for some ξ in N, r, t in T^*, and s in $(T - \{®\})^*$ with $|s| \geq q$. Then s can be pumped according to the Pumping lemma for context-free languages, i.e., $s = s_1s_2s_3s_4s_5$, $|s_2s_3s_4| \leq q_1 \leq q$, s_2s_4 is nonempty, and $rs_1s_2^is_3s_4^is_5t$ is in $L_{CF}(G')$ for all i in $\mathbb{N}$. This situation is depicted in Figure 1.

[1] If the Pumping lemma is stated with two constants, q_1 is the maximum of these two constants. Furthermore, we assume that the same constant applies to all grammars obtained from G' by changing the initial nonterminal. Again, this is easily assured by taking the maximum.

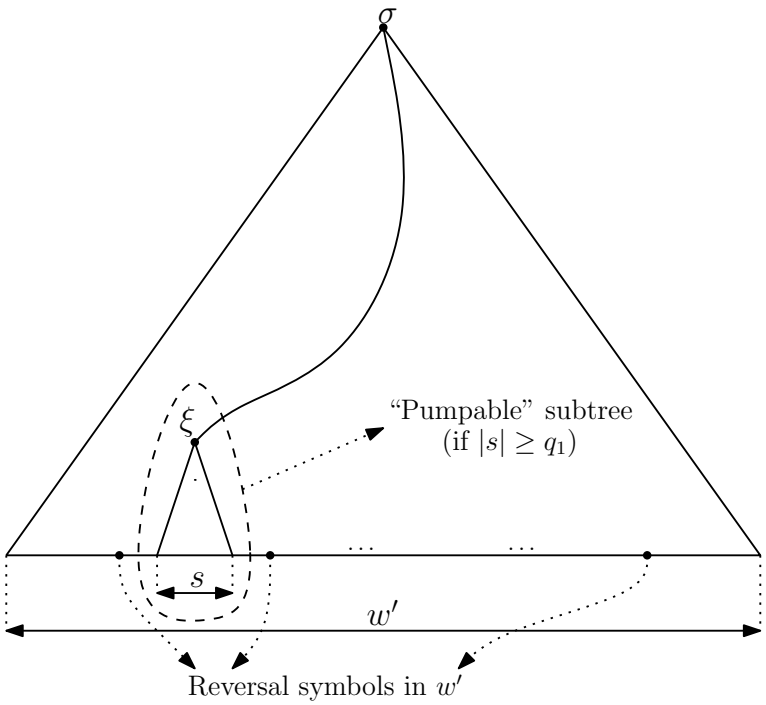

FIGURE 1. If a non-reversal-generating nonterminal ξ produces at least q_1 terminals, the corresponding subtree can be pumped according to the Pumping lemma for context-free languages.

As ξ is non-reversal-generating, s does not contain any reversal symbol, which means that either $w = r's_1s_2s_3s_4s_5t'$ for some r', t' and $r's_1s_2^is_3s_4^is_5t'$ is in L for all i in $\mathbb{N}$, or $w = r''s_5^Rs_4^Rs_3^Rs_2^Rs_1^Rt''$ for some r'', t'' and $r''s_5^R(s_4^R)^is_3^R(s_2^R)^is_1^Rt''$ is in L for all i in $\mathbb{N}$.

It is obvious that in the first case we have $|s_2s_3s_4| \leq q$ and that in the second case we have $|s_4^Rs_3^Rs_2^R| \leq q$. This means that the conditions from the statement of our lemma are satisfied either with $x = r's_1$, $u = s_2$, $y = s_3$, $v = s_4$, $z = s_5t'$, or with $x = r''s_5^R$, $u = s_4^R$, $y = s_3^R$, $v = s_2^R$, $z = s_1^Rt''$.

Let us now consider the remaining case, in which there is no such nonterminal ξ in the derivation tree R'. This means that if a non-reversal-generating nonterminal occurs in the tree, then it has less than $q_1 \leq q$ terminal descendants.

Under this assumption, let us define a reversal-generating context-free grammar $G = (N, T, P, \sigma, ®)$ as follows:

(1) The set of terminals T, the initial nonterminal σ, and the reversal symbol ® are the same as for G'.
(2) The set of nonterminals N consists precisely of the *reversal-generating* nonterminals from N' (and as $k \geq 1$, σ is in N).

(3) The set of production rules P is obtained from P' by deleting production rules from *non-reversal-generating* nonterminals, and by replacing *non-reversal-generating* nonterminals on the right hand sides of the rest of the rules by all possible terminal words shorter than q_1, derivable in G' from the original non-reversal-generating nonterminal.

Formally, let $\eta \to u_1\xi_1u_2\dots\xi_ju_{j+1}$ be a production rule from the original set P', such that η is a *reversal-generating* nonterminal from N', $\xi_1,\dots,\xi_n$ are *non-reversal-generating* nonterminals from N', and words $u_1,\dots,u_{j+1}$ consist solely of terminal symbols and *reversal-generating* nonterminals from N'. Moreover, let $x_1,\dots,x_j$ be words in T^*, such that $|x_1|,\dots,|x_j| < q_1$ and $\xi_i \Rightarrow^*_{G'} x_i$ for $i = 1,\dots,j$ – clearly, the set of all such $x_1,\dots,x_j$ is finite. Then P contains the production rule $\eta \to u_1x_1u_2\dots x_ju_{j+1}$. The set P contains no other production rules.

Obviously, $L_{CF}(G) \subseteq L_{CF}(G')$ and $L(G) \subseteq L(G') = L$. Moreover, it is clear that w' is in $L_{CF}(G)$ – hence w is in $L(G)$. This altogether means that it suffices to "prove the lemma for $L(G)$ instead of L". The grammar G clearly is in the Reversal-aware normal form as well, and contains only reversal-generating nonterminals.

In order to simplify the analysis, we shall assume that the right hand side of each production rule of G either consists of at most one terminal and at most one nonterminal, or consists only of nonterminals. This normal form can easily be achieved by introducing new *reversal-generating* nonterminals and replacing production rules by "chains" of new production rules. Obviously, all properties of G listed above remain unchanged under this transformation.

So let us focus on the grammar G and fix a derivation tree R of w' (that is, w without interpreted reversal symbols) in G. In the rest of the proof, we shall assume that the number k of reversal symbols generated is odd – in other words, $k+1 = 2n$ for some n in $\mathbb{N}$. The proof for k even is similar.

By Proposition 2.2, the word w' generated by G viewed as a context-free grammar, and the word w obtained by the interpretation of reversal symbols in w', may be written as

$$w' = w_1®w_2®\dots®w_n®w_{n+1}®\dots®w_{2n} \rightsquigarrow w_1w_{2n}^Rw_2w_{2n-1}^R\dots w_nw_{n+1}^R = w$$

for some $w_1,\dots,w_{2n}$ in $(T - \{®\})^*$. The key to the proof is to inspect which production rules have lead to the production of particular symbols occurring in $w_1,\dots,w_{2n}$, while using the fact that G is in the Reversal-aware normal form and has only reversal-generating nonterminals.

Let us first introduce some notation. The functions $\zeta_1,\zeta_2 : N \to \{1,\dots,k\}$ assign to a given nonterminal ξ the index of the leftmost and the rightmost reversal symbol generated by ξ, respectively. For $i = 1,\dots,2n$, the *annotation* of w_i is a word $A(w_i)$ in N^*, such that the symbol on the j-th position of $A(w_i)$, denoted by $A(w_i)[j]$, is ξ if and only if the symbol $w_i[j]$ has been (directly) produced, in the derivation tree R, by some production rule from the nonterminal ξ. This notation extends to subwords of w_i as well.

Now, let us make some observations, largely based on elementary properties of derivation trees:

1. For $j = 1, \ldots, |w_1|$, we have $\zeta_1(A(w_1)[j]) = 1$.
2. For $j = 1, \ldots, |w_{2n}|$, we have $\zeta_2(A(w_{2n})[j]) = k$.

The situation described in Observations 1 and 2 is depicted in Figure 2.

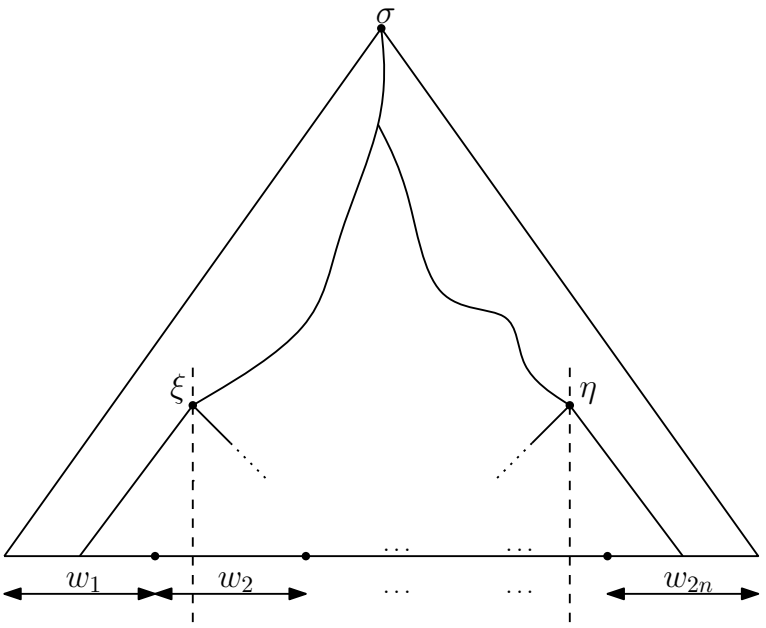

FIGURE 2. Each terminal symbol of w_1 has to be produced from some reversal-generating nonterminal ξ. Necessarily, $\zeta_1(\xi) = 1$. Similarly for the terminal symbols of w_{2n} (recall that $k+1 = 2n$).

3. For $i = 2, \ldots, 2n-1$, the word w_i can be factorized into two possibly empty factors, $w_i = y_i z_i$, so that $\zeta_2(A(y_i)[j]) = i - 1$ for $j = 1, \ldots, |y_i|$ and $\zeta_1(A(z_i)[j]) = i$ for $j = 1, \ldots, |z_i|$.
4. In addition to the observation above, ζ_1 is always less than or equal to $i-1$ on each $A(y_i)$, and ζ_2 is always greater than or equal to i and decreasing on each $A(z_i)$ (but it is not globally decreasing).

The situation described in Observations 3 and 4 is depicted in Figure 3.

5. If $A(w_i)[j] = A(w_i)[j']$ for some i and $j < j'$, then either the subword $w_i[j] \ldots w_i[j'-1]$ or the subword $w_i[j+1] \ldots w_i[j']$ can be pumped in w', together with some other *associated subword* s of w'. This is because the grammar is in the reversal-aware normal form, has no non-reversal-generating nonterminals, and each production rule generates at most one terminal symbol. Thus, if two distinct terminal symbols $w_i[j]$ and $w_i[j']$

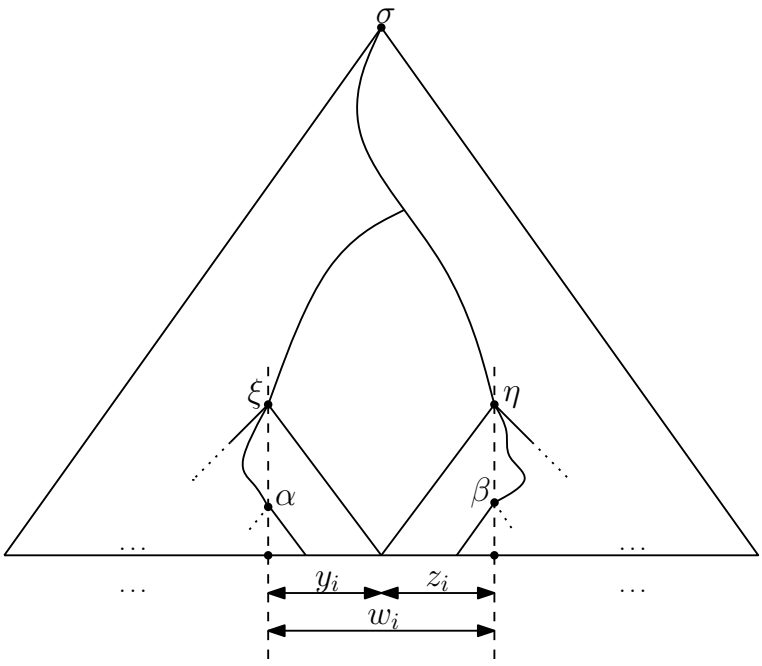

FIGURE 3. According to our assumptions, if a terminal is on the right hand side of a production rule, then this right hand side contains at most one nonterminal. This implies Observation 3. Observation 4 should be clear from the figure.

are generated by the same nonterminal ξ, it means that there is an ancestral chain in the tree R, beginning and ending in ξ, with $w_i[j]$ being the child of the upper ξ and $w_i[j']$ being the child of the lower ξ, or vice versa. In the first case, the subword $w_i[j] \dots w_i[j'-1]$ can be pumped and in the second case, the subword $w_i[j+1] \dots w_i[j']$ can be pumped, since the ancestral chain can be arbitrarily repeated. Returning to observations 1–3, the first case occurs if and only if $i = 1$ or both $w_i[j]$ and $w_i[j']$ belong to z_i, and the second case occurs if and only if $i = 2n$ or both $w_i[j]$ and $w_i[j']$ belong to y_i. The main aim in what follows is to make the associated subword s to be furthest possible from $w_i[j] \dots w_i[j'-1]$ resp. $w_i[j+1] \dots w_i[j']$ after interpreting the reversal symbols. Now, one should see that obtaining the gap of $\frac{|w|}{2k} - q$ would essentially prove the Pumping lemma. In the sequel, we shall call words $w_i[j] \dots w_i[j'-1]$ (resp. $w_i[j+1] \dots w_i[j']$) as above *pumpable subwords*, although possibly some other subwords may be pumped as well.

The situation described in Observation 5 is depicted in Figure 4.

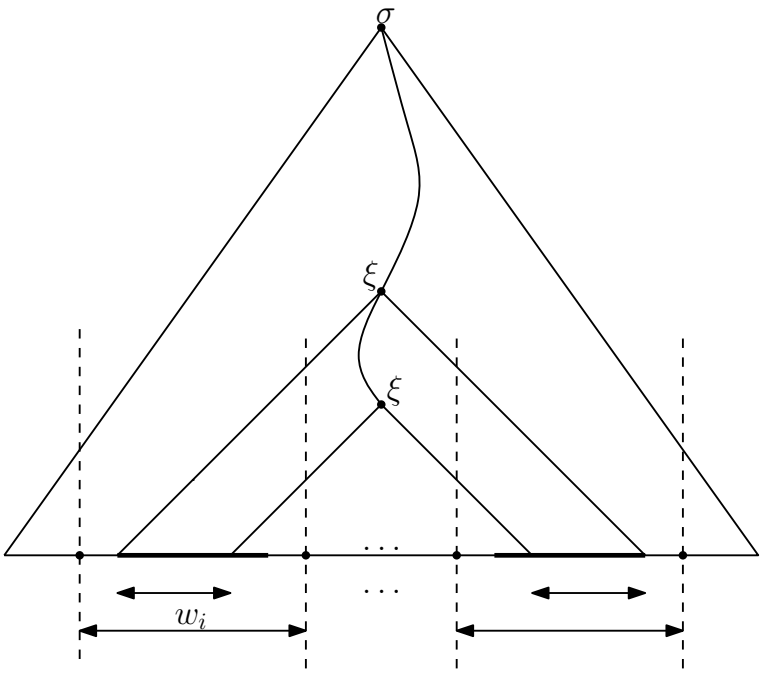

FIGURE 4. A pumpable subword of w_i and its associated subword.

We are now prepared to make the key observation. Recall that for the interpretation of reversal symbols in w', we have

$$w' = w_1 ® w_2 ® \dots ® w_n ® w_{n+1} ® \dots ® w_{2n} \rightsquigarrow w_1 w_{2n}^R w_2 w_{2n-1}^R \dots w_n w_{n+1}^R = w.$$

As w_1 appears at the beginning of w, all pumpable subwords in w_1 have their associated subwords on the right of w_1 (in w, i.e., after interpreting the reversal symbols). Similarly, w_{n+1} appears (reversed) at the end of w, so all pumpable subwords in w_{n+1} have their associated subwords in w on the left of w_{n+1}^R.

Now, take $w_i = y_i z_i$ for some i, $2 \leq i \leq n$. It is a direct corollary of Observation 3 above that all pumpable subwords of y_i have their associated subwords in some of the subwords $w_1, \dots, w_{i-1}$. All these subwords are on the left of w_i after interpreting the reversal symbols. Again by Observation 3, all pumpable subwords of z_i have their associated subwords in some of the subwords $w_{i+1}, \dots, w_{2n}$. Moreover, it follows from Observation 4 that for all j_1, j_2, such that $i+1 \leq j_1 < j_2 \leq 2n$, the pumpable subwords of z_i having their associated subwords in w_{j_2} appear before the pumpable subwords of z_i having their associated subwords in w_{j_1}. In particular, pumpable subwords of z_i having their associated subwords in w on the left of w_i appear before pumpable subwords of z_i having their associated subwords on the right of w_i. To sum up, the word w_i can be divided as $w_i = l_i r_i$, where all

pumpable subwords of l_i have, after the interpretation of reversal symbols, their associated subwords on the left of w_i, and all pumpable subwords of r_i have their associated subwords on the right of w_i.

A symmetrical reasoning can be performed for i with $n+2 \leq i \leq 2n$, resulting in a factorization $w_i^R = l_i^R r_i^R$, where all pumpable subwords of l_i^R have, after the interpretation of reversal symbols, their associated subwords on the left of w_i^R, and all pumpable subwords of r_i^R have their associated subwords on the right of w_i^R. Moreover, in accordance with what has been observed above, we shall denote $r_1 = w_1$ and $l_{n+1}^R = w_{n+1}^R$.

Thus, the word w can be written as

$$w = r_1 l_{2n}^R r_{2n}^R l_2 r_2 l_{2n-1}^R r_{2n-1}^R \dots l_n r_n l_{n+1}^R,$$

with the word w being factored into exactly $2(2n) - 2 = 2k$ subwords. By the Pigeonhole principle, at least one of these subwords is of length at least $\frac{|w|}{2k}$. Suppose that this subword is l_i for some i, $1 \leq i \leq n$ (the reasoning for the case that this subword is r_i, l_i^R or r_i^R is absolutely identical). Recall the definition of pumpable subwords given above. As there are only finitely many nonterminals in N, it follows by the Pigeonhole principle that a constant q_2 (dependent only on the grammar G) exists, such that there has to be a pumpable subword v' in the suffix of l_i of length q_2, provided $\frac{|w|}{2k} \geq q_2$. As v' is a subword of a suffix of length q_2, $|v'| \leq q_2$. Now, let us take a look at the associated subword u' of v'. This may be arbitrarily long and appears on the left of v'. If $|u'| \leq q_2$, let us denote $u = u'$ and $v = v'$. If $|u'| > q_2$, then it follows by the Pigeonhole principle that u' contains a pumpable subword u of length at most q_2. It should be clear that the associated subword of u – let us denote it by v – lies within v'.

Thus, in both cases we have two subwords, u and v of w, yielding a factorization

$$w = xuyvz.$$

It follows that $|uyv| \geq |l_i| - q_2 \geq \frac{|w|}{2k} - q_2$ and $|uv| \leq 2q_2$. Furthermore, we have already mentioned the condition $\frac{|w|}{2k} \geq q_2$, which implies that the reasoning is valid only for w, such that $|w| \geq 2kq_2$. Thus, the Pumping lemma is proved for $q = \max\{q_1, 2kq_2\}$. □

Remark 3.2. In the Introduction, we have claimed that our Pumping lemma for flip-pushdown languages is a generalisation of the classical Pumping lemma for context-free languages. This assertion may be viewed as slightly problematic, as the Pumping lemma for flip-pushdown languages only holds for $\mathscr{L}(\mathrm{NFPDA}_k)$ with $k \geq 1$, while the family of context-free languages identifies with $\mathscr{L}(\mathrm{NFPDA}_0)$ (and it is known that $\mathscr{L}(\mathrm{NFPDA}_i) \subsetneq \mathscr{L}(\mathrm{NFPDA}_{i+1})$ for all i in $\mathbb{N}$). However, setting $k = 0$, the term $\frac{|w|}{2k} - q$ intuitively becomes ∞, thus the condition $|uyv| \geq \frac{|w|}{2k} - q$ cannot be satisfied and might be omitted. In this way we obtain the Pumping lemma for context-free languages, which justifies our claim.

Remark 3.3. For all $k \in \mathbb{N}$, $k \geq 1$, the language L_k defined by

$$L_k = \{\#w_1\$w_1\#w_2\$w_2\# \dots \#w_k\$w_k\# \mid w_1, \dots, w_k \in \{a,b\}^*\}$$

is in $\mathscr{L}(\text{NFPDA}_k)$ [11]. Intuitively, it is obvious that the only possible way of pumping words w from L_k is to pump two occurrences of some w_i, while both occurrences have to be pumped "approximately at the same position", so the distance between both pumped factors is approximately the length of w_i. If the words w_i are of the same length for all i, then the length of each w_i is approximately $\frac{|w|}{2k}$. As a consequence, the bound $|uyv| \geq \frac{|w|}{2k} - q$ of our Pumping lemma cannot be improved by more than a constant.

4. Example Applications

In the proof of their Hierarchy theorem [11], Holzer and Kutrib have considered languages of the following form, defined for all positive integers k:

$$L_k = \{\#w_1\$w_1\#w_2\$w_2\# \dots \#w_k\$w_k\# \mid w_1, \dots, w_k \in \{a,b\}^*\}.$$

As they have observed, the language L_{k+1} is in $\mathscr{L}(\text{NFPDA}_{k+1})$, but is not in $\mathscr{L}(\text{NFPDA}_k)$. The original proof of the latter result is somewhat complicated – it is first argued that if L_{k+1} is in $\mathscr{L}(\text{NFPDA}_k)$, then one can transform L_{k+1} to a context-free language by applying exactly k times the Flip-pushdown input-reversal theorem [11] (reproduced as Theorem 1.6 in the present paper). Next, a generalized Ogden's lemma of Bader and Moura [1] is applied to this context-free language, and finally the Flip-pushdown input-reversal theorem is undone.

In what follows, we shall use our Pumping lemma to obtain the same result. Our proof shall be significantly simpler than the original one by Holzer and Kutrib [11].

Example 4.1. In order to demonstrate the main idea, we shall first prove that the language

$$L_2 = \{\#w_1\$w_1\#w_2\$w_2\# \mid w_1, w_2 \in \{a,b\}^*\}.$$

is not in $\mathscr{L}(\text{NFPDA}_1)$. We shall tackle the general case later in Example 4.2.

Suppose that L_2 is in $\mathscr{L}(\text{NFPDA}_1)$. Let q be a constant guaranteed for L_2 by the Pumping lemma for languages in $\mathscr{L}(\text{NFPDA}_1)$.

Without loss of generality, assume $q > 0$ and take

$$w = \#a^{2q}b^{2q}\$a^{2q}b^{2q}\#a^{2q}b^{2q}\$a^{2q}b^{2q}\#.$$

Then, by the Pumping lemma, words x, u, y, v, z can be found, such that the properties (1)–(4) are satisfied. By the second condition, either $|uyv| \leq q$, or $|uyv| \geq \frac{|w|}{2} - q$ with $|uv| \leq q$. It can be easily verified that the case $|uyv| \leq q$ immediately leads to a contradiction, as the language L_2 is not context-free.

Let us suppose that $|uyv| \geq \frac{|w|}{2} - q$. In other words,

$$|uyv| \geq \frac{|w|}{2} - q = \frac{16q+5}{2} - q > 7q + 2.$$

As $|uv| \leq q$, we obtain

$$|uy| > 6q + 2.$$

It should be clear that in order for xu^2yv^2z to be in L_2, u and v have to be both nonempty and their first letters have to be the same. Moreover, neither u nor v may contain # or \$.

Given $|uy| > 6q + 2$, the above requirements are satisfied only if y contains the middle # – that is, if u belongs to the first half of w, and v belongs to the second half of w. However, the contradiction is immediate in this case.

Example 4.2. Let $k \geq 1$. We shall prove that the language

$$L_{k+1} = \{\#w_1\$w_1\#w_2\$w_2\# \dots \#w_{k+1}\$w_{k+1}\# \mid w_1, \dots, w_{k+1} \in \{a, b\}^*\}$$

is not in $\mathscr{L}(\text{NFPDA}_k)$.

Suppose that L_{k+1} is in $\mathscr{L}(\text{NFPDA}_k)$. Let q be a constant guaranteed for L_{k+1} by the Pumping lemma for languages in $\mathscr{L}(\text{NFPDA}_k)$.

Without loss of generality, we may assume $q > 0$. Let us set

$$w_1 = w_2 = \ldots = w_{k+1} = \left(a^{2q}b^{2q}\right)^k$$

and take

$$w = \#w_1\$w_1\#w_2\$w_2\# \dots \#w_{k+1}\$w_{k+1}\#.$$

Clearly, w is in L_{k+1}, so the Pumping lemma implies the existence of words x, u, y, v, z, such that the properties (1)–(4) are satisfied. By the second condition, either $|uyv| \leq q$, or $|uyv| \geq \frac{|w|}{2k} - q$ with $|uv| \leq q$. The case $|uyv| \leq q$ immediately leads to a contradiction, as the language L_{k+1} is not context-free.

For this reason, we may suppose that $|uyv| \geq \frac{|w|}{2k} - q$. In other words,

$$|uyv| \geq \frac{|w|}{2k} - q = \frac{8k(k+1)q + 2k + 3}{2k} - q > (4k+3)q + 1.$$

As $|uv| \leq q$, we obtain

$$|uy| > (4k+2)q + 1.$$

In order for xu^2yv^2z to be in L_{k+1}, u and v have to be both nonempty and their first letters have to be the same. Moreover, neither u nor v may contain # or \$.

If uyv is not contained in a subword $w_i\$w_i$ for some i, then the contradiction is immediate. So let us suppose that uyv *is* contained in such a subword.

The word w_i is, by definition, composed of k "blocks" of the form $a^{2q}b^{2q}$, and the total length of w_i is $4kq$. Given $|uy| > (4k+2)q + 1$, the above requirements can be satisfied only if u does not begin in the same block of w_i (in the first

occurrence of w_i) as v (in the second occurrence of w_i). The contradiction follows easily.

Example 4.3. As a last example, we shall prove that the language

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

is not in $\mathscr{L}(\text{NFPDA}_{fin})$. This fact is well known and can easily be proved using alternative methods [11]. However, our aim is to demonstrate that the Pumping lemma for flip-pushdown languages can be utilized to prove that a language is not in $\mathscr{L}(\text{NFPDA}_{fin})$.

Suppose that L is in $\mathscr{L}(\text{NFPDA}_k)$ for some $k \geq 1$. Let q be a constant guaranteed for L by the Pumping lemma.

Take $w = a^q b^q c^q$ in L. It follows that there are words x, u, y, v, z, such that the properties (1)–(4) of the Pumping lemma are satisfied for w. It can be easily seen that if u or v contains two or more different symbols, then xu^2yv^2z does not belong to $a^*b^*c^*$, and hence is not in L. On the other hand, if $u \in e^*$ and $v \in f^*$ for some e and f in $\{a, b, c\}$, then clearly $xu^2yv^2z = a^i b^j c^k$, where either $i \neq j$, or $i \neq k$. Thus, xu^2yv^2z is not in L and a contradiction follows.

It should be noted that the bound $|uyv| \geq \frac{|w|}{2k} - q$ is irrelevant when proving that a language is not in $\mathscr{L}(\text{NFPDA}_{fin})$, as k can be arbitrarily large.

Conclusion

We have formulated and proved a pumping lemma for the families of languages accepted by flip-pushdown automata with a number of flips limited by a constant. The lemma provides an efficient tool for disproving existence of flip-pushdown automata for specific languages. In particular, it is possible to apply the lemma to prove that a given language does not belong to $\mathscr{L}(\text{NFPDA}_k)$ for some specific k or that the language does not belong to $\mathscr{L}(\text{NFPDA}_{fin})$. As demonstrated by the examples of Section 4, in particular by Example 4.2, proofs using our pumping lemma might be significantly simpler compared to proofs via previously known techniques.

In addition to proving the result for its own sake, the techniques used have hopefully convinced the reader about the power of the grammatical viewpoint on flip-pushdown automata.

In [14], the present author has described two families of grammars, which are (after some suitable restriction) equivalent to flip-pushdown automata with a constant number of flips: *reversal-generating context-free grammars* (RGCFG) and *parallel interleaving grammar systems* (PIGS). Holzer and Kutrib [11] have asked a question regarding the relation of flip-pushdown automata with a constant number of flips to E0L systems and to ET0L systems. The former question has already been successfully settled by Ďuriš and Košta [7], who have proved that the families $\mathscr{L}(\text{NFPDA}_{fin})$ and $\mathscr{L}(\text{E0L})$ are incomparable. As an example application of parallel interleaving grammar systems, it has been proved in [14] that flip-pushdown

automata with a constant number of flips are strictly weaker than ET0L systems, which has settled the latter problem.

The characterisation in terms of reversal-generating context-free grammars has only been used in [14] to prove the characterisation in terms of PIGS. That is, the pumping lemma presented in this paper is the first *direct* application of reversal-generating grammars to the theory of flip-pushdown languages.

Of course, the presented pumping lemma is not the only possible application of RGCFGs that naturally comes to mind. For instance, Ďuriš and Košta have obtained a relatively technical proof [6] of the fact that spontaneous transitions in flip-pushdown automata with a constant number of flips can be removed (without affecting the number of flips). The proof via reversal-generating grammars is considerably simpler: as every reversal-generating context-free grammar G can be viewed as an "ordinary" context-free grammar, the result can be established relatively easily by using the fact that G can be transformed into the Double Greibach normal form [9, 15].

In a similar spirit, the simulation of reversal-generating grammars by flip-pushdown automata (described in the proof of Lemma 2.9) solves in affirmative an open problem of Ďuriš and Košta [6], who have asked if each language from $\mathscr{L}(\mathrm{NFPDA}_{fin})$ can be accepted by a flip-pushdown automaton with a constant number of states (that is independent of the number of flips).

The author also believes that reversal-generating context-free grammars may be used to unify certain concepts. In particular, Bordihn, Holzer, and Kutrib have introduced so called *input-reversal pushdown automata* [3], a model in some sense symmetric to flip-pushdown automata, for which they have proved its equivalence to linear indexed grammars [8]. It should be immediate that input-reversal pushdown automata are equivalent to a modification of RGCFGs, in which reversal symbols are interpreted in the right-to-left order. This makes the symmetry between input-reversal and flip-pushdown automata even more evident.

Finally, it seems likely that a RGCFG-like characterisation may be obtained for several other families of automata, such as *hairpin automata* of Bordihn, Holzer, and Kutrib [4].

References

[1] C. Bader and A. Moura. A generalization of Ogden's lemma. *J. ACM*, 29(2):404–407, 1982.
[2] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Z. Phonetik. Sprachwiss. Kommunikationsforsch.*, 14(2):143–172, 1961.
[3] H. Bordihn, M. Holzer, and M. Kutrib. Input reversals and iterated pushdown automata: A new characterization of Khabbaz geometric hierarchy of languages. In C. C. Calude, E. Calude, and M. J. Dinneen, editors, *DLT 2004*, volume 3340 of *LNCS*, pages 102–113. Springer, 2004.
[4] H. Bordihn, M. Holzer, and M. Kutrib. Hairpin finite automata. *J. Autom. Lang. Comb.*, 16(2–4):71–74, 2011.
[5] N. Chomsky. On certain formal properties of grammars. *Inf. Control*, 2(2):137–167, 1959.
[6] P. Ďuriš and M. Košta. Flip-pushdown automata: nondeterministic ε-moves can be removed. In M. Lopatková, editor, *ITAT 2011*, pages 15–22, 2011.

[7] P. Ďuriš and M. Košta. Flip-pushdown automata with k pushdown reversals and E0L systems are incomparable. *Inf. Process. Lett.*, 114(8):417–420, 2014.
[8] J. Duske and R. Parchmann. Linear indexed languages. *Theor. Comput. Sci.*, 32:47–60, 1984.
[9] J. Engelfriet. An elementary proof of Double Greibach normal form. *Inf. Process. Lett.*, 44(6):291–293, 1992.
[10] S. A. Greibach. A new normal-form theorem for context-free phrase structure grammars. *J. ACM*, 12(1):42–52, 1965.
[11] M. Holzer and M. Kutrib. Flip-pushdown automata: $k + 1$ pushdown reversals are better than k. In J. Baeten et al., editors, *ICALP 2003*, volume 2719 of *LNCS*, pages 490–501. Springer, 2003.
[12] M. Holzer and M. Kutrib. Flip-pushdown automata: Nondeterminism is better than determinism. In Z. Ésik and Z. Fülöp, editors, *DLT 2003*, volume 2710 of *LNCS*, pages 361–372. Springer, 2003.
[13] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson, third edition, 2006.
[14] P. Kostolányi. Two grammatical equivalents of flip-pushdown automata. In G. F. Italiano et al., editors, *SOFSEM 2015*, volume 8939 of *LNCS*, pages 302–313. Springer, 2015.
[15] D. J. Rosenkrantz. Matrix equations and normal forms for context-free grammars. *J. ACM*, 14(3):501–507, 1967.
[16] P. Sarkar. Pushdown automaton with the ability to flip its stack. *Electronic Colloquium on Computational Complexity (ECCC)*, 8, 2001.

Communicated by (The editor will be set by the publisher).
...