A Solution to Yamakami's Problem on Non-uniform Context-free Languages

Toshio Suzuki, *Member, IAENG*

***Abstract*—Yamakami (*Theoret. Comput. Sci.*, 2011) studies non-uniform context-free languages. Here, the length of advice is assumed to be the same as that of an input. Let CFL and CFL/n denote the class of all context-free languages and its non-uniform version, respectively. We let CFL(2) denote the class of intersections of two context-free languages. An interesting direction of a research is asking how complex CFL(2) is, relative to CFL. Yamakami raised a problem whether there is a CFL-immune set in CFL(2) - CFL/n. The best known so far is that LSPACE - CFL/n has a CFL-immune set, where LSPACE denotes the class of languages recognized in logarithmic-space. We present an affirmative solution to his problem. Two key concepts of our proof are the overlapped palindrome and Yamakami's swapping lemma. The swapping lemma is applicable to the setting where the pumping lemma (Bar-Hillel's lemma) does not work. Our proof is an example showing how useful the swapping lemma is. In addition, by means of Kolmogorov complexity, we show the following: With respect to realtime deterministic context-free languages, the non-uniform class with parallel advice is not a subset of that with serial advice.**

***Index Terms*—context-free language; pushdown automaton; advice function; non-uniform complexity class; immune set.**

I. Introduction

THE regular languages have beautiful closure properties. For example, given two regular languages, their intersection is a regular language. Nevertheless, in the studies of programming languages, most of important languages are not regular. The same holds in the studies of formal models of natural languages. In the case of classes larger than the regular languages, closure properties are more difficult than the regular cases.

In particular, given two context-free languages, their intersection is not necessarily context free. For a positive integer k, we consider the intersection of k context-free languages, and let CFL(k) denote the class of all such intersections. CFL(1) is CFL, the class of all context-free languages. It is known that CFL(k) is a proper subset of CFL($k+1$).

How complex is CFL($k+1$), relative to CFL(k)? An interesting observation is given by Flajolet and Steyaert [6]. Let $L_{3\text{eq}}$ denote the set of all strings of the form $0^n 1^n 2^n$, where n is a natural number. It is easily seen that $L_{3\text{eq}}$ belongs to CFL(2).

An immune set is a key concept in the classical recursion theory, namely in Post's problem. Later, immune sets relative to complexity classes are studied in the complexity theory [18]. Given a class $\mathcal{C}$ of languages, an infinite language A is $\mathcal{C}$*-immune* if no infinite subset of A belongs to $\mathcal{C}$. Flajolet and Steyaert observed that $L_{3\text{eq}}$ is CFL-immune. Being $\mathcal{C}$-immune is a much stronger condition than non-membership in $\mathcal{C}$. Thus, the above observation shows that $L_{3\text{eq}}$, a member of CFL(2), is far from belonging to CFL.

We consider another condition stronger than non-membership in a given class $\mathcal{C}$. It is the non-membership in a non-uniform version of $\mathcal{C}$. In the usual mathematical model of computation, a fixed algorithm processes all the inputs. Such a type of computation is called *uniform*. In the case where the inputs are classified according to a certain parameter, typically the input size, and a hardware is assigned to each parameter, we need a stronger model. Such a type of computation is called *non-uniform*. A typical case is given by a family of circuits. Here, the family is not necessarily computable. Non-uniform computation has meaningful applications. For example, in the studies of cryptography, non-uniform computation plays a role of a powerful adversary [5].

A nice formulation of non-uniform computation is achieved by a computation with an advice function. An advice function is a function of a natural number to a string. An advice function is not necessarily computable. With an input, the advice at the length of the input is provided to a fixed algorithm. Pippenger [13] characterizes computational power of polynomial-sized circuits by advice functions. Karp and Lipton [9] establish the foundation of computation with an advice function. Roughly speaking, given a class $\mathcal{C}$ of languages and a size-bound β for advice functions, a non-uniform class $\mathcal{C}/\beta$ is defined. In most cases, even if all the members of $\mathcal{C}$ are computable languages, $\mathcal{C}/\beta$ contains non-computable languages. Thus, $\mathcal{C}/\beta$ is much larger than $\mathcal{C}$. Damm and Holzer [4] investigate, in the style of Karp and Lipton, finite automata that take advice. In section IV, we shall review their definitions.

Tadaki et al. [15] investigate computation with advice in slightly different style from that of Karp and Lipton. Given a class $\mathcal{C}$ of languages, we define $\mathcal{C}/n$ as follows. Suppose that L is a language over an alphabet Σ. Suppose that Γ is another alphabet. We introduce an extended alphabet $\begin{bmatrix} \Sigma \\ \Gamma \end{bmatrix}$. It consists of all symbols of the form $\begin{bmatrix} x \\ a \end{bmatrix}$ for $x \in \Sigma$ and $a \in \Gamma$. Given two strings $x = x_1 \cdots x_n \in \Sigma^n$ and $a = a_1 \cdots a_n \in \Gamma^n$ of the same length, we let $\begin{bmatrix} x \\ a \end{bmatrix}$ denote the string $\begin{bmatrix} x_1 \\ a_1 \end{bmatrix} \cdots \begin{bmatrix} x_n \\ a_n \end{bmatrix} \in \begin{bmatrix} \Sigma \\ \Gamma \end{bmatrix}^n$.

Definition 1. (Tadaki et al. [15]) A language L belongs to $\mathcal{C}/n$ if and only if there exist a language $L' \in \mathcal{C}$ and a function $h : \mathbb{N} \to \Gamma^*$ such that for every $x \in \Sigma^*$, the length

Manuscript received September 13, 2015; revised January 14 and February 7, 2016. This work was partially supported by Japan Society for the Promotion of Science (JSPS) KAKENHI (C) 22540146.

Department of Mathematics and Information Sciences, Tokyo Metropolitan University, Minami-Ohsawa, Hachioji, Tokyo 192-0397, Japan. e-mail: toshio-suzuki@tmu.ac.jp (see http://researchmap.jp/read0021048/?lang=english).

of $h(|x|)$ is the same as that of x and the following holds.

$$x \in L \;\Leftrightarrow\; \begin{bmatrix} x \\ h(|x|) \end{bmatrix} \in L'$$

Then, h is called an *advice function*. $h(n)$ is the advice at length n. $\square$

Here, the n of CFL/n denotes that the length of advice is same as the input length. The following are examples on DCFL, the class of languages accepted by deterministic pushdown automata, and REG, the regular languages. In section IV, we will review more precise definition of DCFL.

(i) Let L_{eq} denote the set of all strings of the form 0^n1^n, where n is a natural number. Then L_{eq} belongs to DCFL $\cap$ REG$/n$ and is REG-immune [6].

(ii) Let Pal$_\#$ denote the palindromes whose center symbol is the separator symbol $\#$. Then Pal$_\#$ belongs to DCFL $-$ REG$/n$ and is REG-immune [17].

The CFL-immune set $L_{3\text{eq}}$ given in [6] belongs to CFL(2) $\cap$ CFL$/n$; in order to verify that $L_{3\text{eq}} \in$ CFL$/n$, consider an advice function $h(3n) = 0^n1^n2^n$. Thus, it is natural and interesting to ask whether there is a CFL-immune set in CFL(2) $-$ CFL$/n$.

Yamakami's problem Yamakami [17] raised a problem whether there is a CFL-immune set in CFL(2) - CFL/n.

The best known so far is that LSPACE - CFL/n has a CFL-immune set [17], where LSPACE denotes the class of languages recognized by deterministic Turing machines with a single read-only input tape and a logarithmic-space bounded work tape.

What is the difficult point in the problem of Yamakami? A classical method of showing that a language is not context free is the pumping lemma for CFL (Bar-Hillel's lemma [2]). However, the pumping lemma destroys the advice $h(n)$.

Our main theorem is an affirmative solution to the problem of Yamakami. Two key concepts of our proof are the overlapped palindrome and Yamakami's swapping lemma. The swapping lemma is applicable to the setting where the pumping lemma does not work. Our proof is an example showing how useful the swapping lemma is. We show our main theorem in section III.

In the setting of Damm and Holzer [4], the arrangement of the advice and the input is serial. In section IV, we observe that the same result as our main theorem holds for the advised language class of Damm and Holzer.

In the remainder of Section IV, we show some separation results between non-uniform classes with parallel-advice and those with serial advice. Among others, in the case of realtime deterministic context-free languages, we show that the parallel class is not a subset of the serial class. Our main tool is Kolmogorov complexity. Finally, we present some open problems.

II. Preliminaries

A. Notation

For two sets A and B, their *difference* $A - B$ is $\{x \in A : x \notin B\}$. $A \subset B$ denotes that A is a subset of B; A may equal to B. $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of all natural numbers. For a real number x, $\lceil x \rceil$ denotes the minimal natural number $n \geq x$.

An *alphabet* denotes a finite set of characters. For an alphabet Σ, the set of all strings is denoted by Σ^*. The set of all non-empty strings is denoted by Σ^+. Given a string w, its *length* $|w|$ denotes the total number of occurrences of characters. The *reverse* of $w = w_1 \cdots w_n$, where $n = |w|$, is $w_n \cdots w_1$. The reverse of w is denoted by w^R.

REG (CFL, respectively) is the class of all regular (context-free) languages [8]. Suppose that $\mathcal{C}$ is a given class of languages such as REG or CFL. An advised class $\mathcal{C}/n$ is defined in the introduction.

When we discuss acceptance of an input by a pushdown automaton, we consider acceptance by final state. During the computation, the head has to scan all the letters of the input string. At the end of computation, the state is a final state. We allow non-empty stack at the end of computation.

The class CFL/n is characterized by *non-deterministic pushdown automata with an advice function* (Fig. 1). It has a one-way read-only input tape and a pushdown memory (stack). The input tape has two tracks. An input is given on the first track. The advice at the length of the input is given on the second track. Then the automaton works as a non-deterministic automaton over the alphabet $\begin{bmatrix} \Sigma \\ \Gamma \end{bmatrix}$.

In Fig. 1, $z_k \cdots z_1 \perp$ is the string in the pushdown memory (stack). $\perp$ denotes the symbol denoting the bottom of the stack. ¢ is the left-end symbol. \$ is the right-end symbol. $h(n) = a_1 \cdots a_n$ in the second track is the advice at length n. The head is reading a symbol $\begin{bmatrix} x_1 \\ a_1 \end{bmatrix}$. q_0 is the current state.

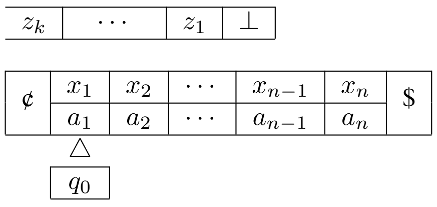


Fig. 1. A non-deterministic pushdown automaton with advice

B. The Swapping Lemma for Context-free Languages

Ogden et al. [12] show the interchange lemma for context-free languages. In some situation where the usual pumping lemma does not work, the interchange lemma is a useful tool for showing a given language is not context free. Examples and expositions may be found in [3] and in [14]. The swapping lemma of Yamakami has much in common with the interchange lemma, but they are different results.

Suppose that n is a positive integer, S is a set of strings of length n, i is a natural number, and that u is a string over Σ such that $i + |u| \leq n$. Yamakami [16] defines a subset $S_{i,u}$ of S as follows.

$$S_{i,u} = \{v_1 \cdots v_n \in S : v_{i+1} \cdots v_{i+|u|} = u\}$$

The swapping lemma asserts that if the ratio $|S_{i,u}|/|S|$ is small enough for any i and u (with certain properties), then there exist two strings $x, y \in S$ such that x' and y' belong to L, where strings x' and y' are obtained by swapping the midsections of x and y, and such that the midsections are different.

Lemma 1 (The Swapping Lemma for Context-free Languages [16]). *Suppose that Σ has at least two letters, and that L is an infinite context-free language over Σ. Then, there exists a positive integer m,* a swapping lemma constant, *with the following properties.*

Suppose that $n \geq 2$ is a natural number, S is a subset of $L \cap \Sigma^n$, and j_0, k are natural numbers such that $2 \leq j_0$, $2j_0 \leq k \leq n$, and such that for any positive integer $i \leq n-j_0$ and any string $u \in \Sigma^{j_0}$, we have:

$$|S_{i,u}| < |S|/m(k-j_0+1)(n-j_0+1)$$

Then, there exist positive integers i, j and two strings $x = x_1x_2x_3, y = y_1y_2y_3 \in S$ with the following properties: $i + j \leq n$, $j_0 \leq j \leq k$, $|x_1| = |y_1| = i$, $|x_2| = |y_2| = j$, $|x_3| = |y_3|$, $x_2 \neq y_2$, $x_1y_2x_3 \in L$, and $y_1x_2y_3 \in L$.

III. Main Theorem and Its Proof

Definition 2. We define our test language L_2 as follows (the suffix 2 is that of CFL(2)). Here, $\#$ is a letter that does not belongs to $\{0, 1\}$.

$$L_2 := \{w\#w^R\#w : w \in \{0,1\}^+\}$$

□

Lemma 2. *L_2 belongs to* CFL(2).

Proof: We define languages $L_{2,1}$ and $L_{2,2}$ as follows. These are clearly context-free languages.

$$L_{2,1} := \{w\#w^R\#x : w \in \{0,1\}^+, x \in \{0,1\}^+\}$$
$$L_{2,2} := \{x\#w\#w^R : w \in \{0,1\}^+, x \in \{0,1\}^+\}$$

Then it holds that $L_2 = L_{2,1} \cap L_{2,2}$. Hence L_2 belongs to CFL(2). ■

Lemma 3. *L_2 is* CFL*-immune.*

Proof: A standard argument based on Bar-Hillel's lemma shows that L_2 is CFL-immune. ■

Theorem 4. *(Main theorem) There exists a* CFL*-immune set in* $\mathrm{CFL}(2) - \mathrm{CFL}/n$.

Proof: By Lemmas 2 and 3, it is sufficient to show that L_2 does not belong to CFL/n. We work with Yamakami's swapping lemma for context-free languages [16]. Consult Example 4.2 of [16] for a basic usage of the swapping lemma. For a proof by contradiction, fix a function h and a context-free language L such that $\forall n\ |h(n)| = n$, and such that for any $\xi \in \{0,1\}^*$, the following holds.

$$\xi \in L_2 \ \leftrightarrow\ \begin{bmatrix} \xi \\ h(|\xi|) \end{bmatrix} \in L$$

Let m be a swapping lemma constant for the context-free language L. Let $n > 0$ be a multiple of 16 with the following property.

$$2^{n/4} > (2mn^2)^4 \tag{1}$$

We define a subset S of L as follows.

$$S := \left\{ \begin{bmatrix} \xi \\ h(n) \end{bmatrix} \in L : \xi \in \{0,1\}^n \right\}$$

Since $w\#w^R\#w$ is uniquely determined by $w \in \{0,1\}^+$, the following holds.

$$|S| = 2^{(n-2)/3} \tag{2}$$

Let k and j_0 be the followings. Here, the base of the logarithm is 2.

$$k := n/4 \tag{3}$$
$$j_0 := 2(\lceil \log(mn^2) \rceil + 1) \tag{4}$$

By (1) and (3), $k = n/4 > 4(\log(mn^2) + 1)$. Since $n/4$ is a multiple of 4, $k \geq 4(\lceil \log(mn^2) \rceil + 1)$. By (4), we get the following.

$$k \geq 2j_0 \tag{5}$$

Given a natural number i and a string u over the alphabet of L such that $i + j_0 \leq n$ and $|u| = j_0$, we define $S_{i,u}$ as follows.

$$S_{i,u} = \{v_1 \cdots v_n \in S : v_{i+1} \cdots v_{i+j_0} = u\}$$

In the case where $\begin{bmatrix} w\#w^R\#w \\ h(n) \end{bmatrix}$ is in $S_{i,u}$, some bits of w are bound by u. We are going to estimate the number of bits bound by u. The minimal number is achieved when u spans the border of any two blocks (of the first track) with the center of u at the border. Thus, at least $\lceil (j_0 - 1)/2 \rceil$ bits of w are bound by u. Therefore, we have the following.

$$|S_{i,u}| \leq 2^{-(j_0-1)/2}|S| \tag{6}$$

Now, we have the following.

$$|S_{i,u}| < |S|/kmn \tag{7}$$

This is shown as follows. $2^{(j_0-1)/2} \geq \sqrt{2}mn^2$ [by (4)] $= 4\sqrt{2}kmn$ [by (3)] $> kmn$. Thus, $2^{(j_0-1)/2} > kmn$. By (6), we have shown (7).

By (5) and (7), we can apply the swapping lemma for context-free languages [16, Lemma 4.1] to the present setting.

By the swapping lemma, there exist natural numbers i, j and strings $x, y \in S$ with the following properties.

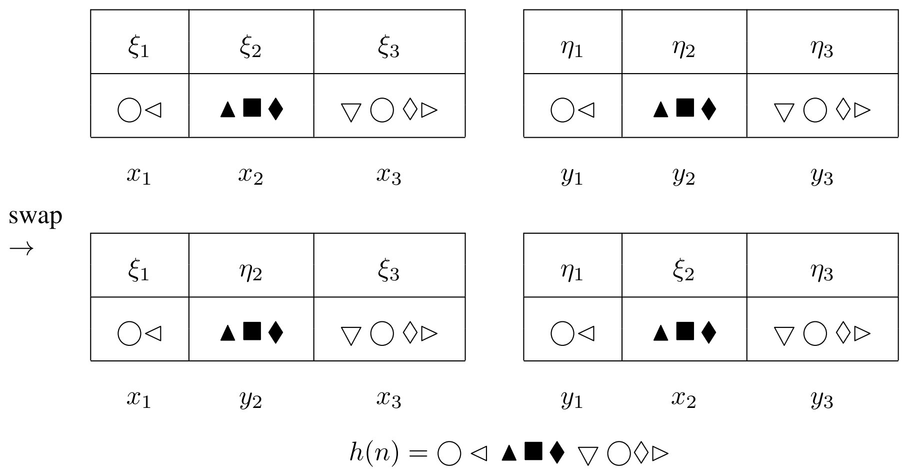
- $1 \leq i \leq n - j$ and $j_0 \leq j \leq k (= n/4)$
- x, y are of the form $x = x_1x_2x_3, y = y_1y_2y_3$, where each x_ℓ and y_ℓ are strings, and it holds that $|x_1| = |y_1| = i$, $|x_2| = |y_2| = j$, $|x_3| = |y_3|$, $x_2 \neq y_2$, $x_1y_2x_3 \in L$ and $y_1x_2y_3 \in L$.

Let ξ, η, ξ_ℓ and η_ℓ ($\ell = 1, 2, 3$) be the projections of x, y, x_ℓ and y_ℓ to the first track, respectively. For example, $x = \begin{bmatrix} \xi \\ h(n) \end{bmatrix}, y = \begin{bmatrix} \eta \\ h(n) \end{bmatrix}$.

Since x belongs to L and the second component is $h(n)$, it holds that $\xi \in L_2$. Therefore ξ is of the form $w\#w^R\#w$ for some string w of length $(n-2)/3$. Here, it holds that $2 < |\xi_2| < |w|$. The first inequality is shown as follows: $2 < j_0$ [by (4)] $\leq j = |\xi_2|$. The second inequality is shown as follows: $|\xi_2| = j \leq k = n/4 < (n-2)/3 = |w|$.

Therefore, ξ_2 is not included by $\#$. ξ_2 is included by one of $w\#$, $w^R\#$ and (the rightmost) w; or by consecutive two of them. The same holds for η_2.

Fig. 2 demonstrates the case of $i + j \leq (n+1)/3$. Here, ξ_2 is included by $w\#$. Fig. 3 demonstrates the case of $i \leq$



ξ_1 ξ_2 ξ_3
η_1 η_2 η_3
x_1 x_2 x_3
y_1 y_2 y_3
swap $\rightarrow$
ξ_1 η_2 ξ_3
η_1 ξ_2 η_3
x_1 y_2 x_3
y_1 x_2 y_3
$h(n) = \bigcirc \triangleleft \blacktriangle \blacksquare \blacklozenge \triangledown \bigcirc \lozenge \triangleright$

Fig. 4. The invariance of the second track under swapping

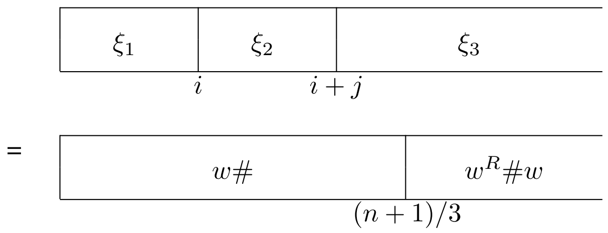


Fig. 2. The case of $i + j \le (n + 1)/3$

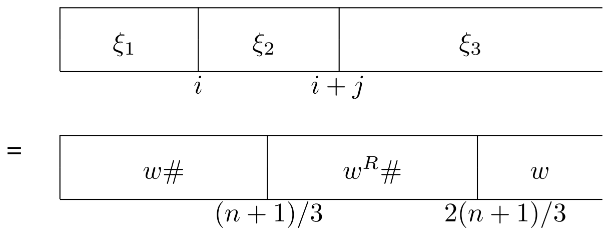


Fig. 3. The case of $i \le (n + 1)/3 < i + j$

$(n + 1)/3 < i + j$. Here, ξ_2 is included by $w\#w^R\#$. The other cases are similar.

The swapping of x_2 and y_2 does not affect the second components $h(n)$ of x and y (Fig. 4). Thus both $\xi_1\xi_2\xi_3$ and $\xi_1\eta_2\xi_3$ belong to L_2, and $\xi_2 \neq \eta_2$. Hence, we get a contradiction.

Thus, we have shown that L_2 does not belong to CFL/n. Hence, we have shown the theorem. ■

IV. Serial Advices

A. A Variation of the Main Theorem

In the definition of Damm and Holzer [4], the arrangement of the advice and the input is serial, while in that of Tadaki et al., it is parallel. In this subsection, we show that the same result as our main theorem holds for the advised language class in the sense of Damm and Holzer.

Definition 3. (Damm and Holzer [4]) Given a class $\mathcal{C}$ of languages, the advised language class $\mathcal{C}/n$ in the sense of Damm and Holzer is defined as follows. Suppose that Σ and Γ_0 are alphabets. Suppose that L is a language over Σ. Then, L belongs to $\mathcal{C}/n$ (in the sense of Damm and Holzer) if and only if there exist a language $L'' \in \mathcal{C}$ and a function $g : \mathbb{N} \rightarrow \Gamma_0^*$ such that $\forall n\ |g(n)| = n$, and such that the following holds.

$$x \in L \ \Leftrightarrow \ g(|x|)\ x \in L''$$

Here, $g(|x|)\ x$ is the concatenation of $g(|x|)$ and x. Then, g is called an *advice function*. $g(n)$ is the advice at length n. □

In the remainder of the paper, $\mathcal{C}/n$ denotes the advised class in the sense of Tadaki, Yamakami and Lin [15], that is, the parallel one defined in Introduction. On the other hand, $(\mathcal{C}/n)_{\mathrm{serial}}$ denotes the advised class defined in this section. The following is a variation of our main theorem. The proof is similar to that of the main theorem.

Theorem 5. *There exists a* CFL*-immune set in* CFL(2) $-$ $(\mathrm{CFL}/n)_{\mathrm{serial}}$.

B. Parallel versus Serial: Finite Automaton

In this subsection, we discuss separation of parallel advice classes and serial advice classes.

In general, the two concepts of advised classes do not coincide. Recall that REG denotes the class of all regular languages.

Example 1. REG/n is not a subset of $(\mathrm{REG}/n)_{\mathrm{serial}}$. A proof is as follows. Damm and Holzer show that the language $L_{\mathrm{eq}} = \{0^n1^n : n \in \mathbb{N}\}$ does not belong to $(\mathrm{REG}/n)_{\mathrm{serial}}$ [4, Propositions 1 and 7]. On the other hand, let $h(n)$ be $0^{n/2}1^{n/2}$ if n is even; 2^n otherwise. Then, by means of the advice function h, L_{eq} is shown to be in REG/n. □

Example 2. $(\mathrm{REG}/n)_{\mathrm{serial}}$ is a subset of REG/n. A proof is as follows. Suppose L is an element of $(\mathrm{REG}/n)_{\mathrm{serial}}$. Let L'' and g be a regular language and an advice function satisfying the requirements in Definition 3. Let M be a deterministic finite automaton that accepts L''. Given a natural number n, let $q_{(n)}$ be the state when M has read $g(n)$. Provided that $q_{(n)}$ is given, without knowing what $g(n)$ is, we can simulate the moves of M after reading $g(n)$. Then, we define $h(n)$ as to be $q_{(n)}\ 0^{n-1}$. The 0s in the tail are just for adjusting the length of $h(n)$. Let Γ be the union of $\{0\}$ and the set of the states of M. Now, it is easy to define a regular language L' satisfying the requirements in

Definition 1 with respect to Γ and h. Therefore, L belongs to REG/n. □

A direct proof of Example 1 is given by means of prefix-free Kolmogorov complexity.

Definition 4. [11]

- A string $u = u_1 \cdots u_m$ is a *prefix* of a string $v = v_1 \cdots v_n$ if $m \leq n$ and for each $i \leq m$ it holds that $u_i = v_i$. A set S of strings is *prefix-free* if for each $u, v \in S$ such that $u \neq v$, u is not a prefix of v. A partial function $M : \{0,1\}^* \to \{0,1\}$ is called a *prefix-free machine* if M is a partial recursive function and the domain of M is a prefix-free set.
- For a prefix-free machine M, its *descriptive complexity* $K_M : \{0,1\}^* \to \mathbb{N} \cup \{\infty\}$ is defined as follows. Suppose $x \in \{0,1\}^*$. If there exists $\sigma \in \{0,1\}^*$ such that $M(\sigma) = x$ then $K_M(x)$ is the length of a shortest such σ. If there is no such σ then $K_M(x)$ is ∞.
- A prefix-free machine R is an *optimal prefix-free machine* if for each prefix-free machine M, there is a constant d (depending on M) such that for each $x \in \{0,1\}^*$, $K_R(x) \leq K_M(x) + d$.

□

It is known that there exists an optimal prefix-free machine [11, Proposition 2.2.7]. We fix such a machine R, and let K denote K_R. For an infinite binary string $Z : \mathbb{N} \to \{0,1\}$ and a natural number n, we let $Z \upharpoonright n$ denote $Z(0)Z(1) \cdots Z(n-1)$. It is known that there exists a binary string Z of the following property [11, section 3.2].

$$\exists b \in \mathbb{N}\ \forall n \in \mathbb{N}\ [K(Z \upharpoonright n) > n - b] \tag{8}$$

Example 3 is a refinement of the proof by Damm and Holzer [4] that $L_{\mathrm{eq}} = \{0^n 1^n : n \in \mathbb{N}\} \notin (\mathrm{REG}/n)_{\mathrm{serial}}$.

Example 3. A direct proof that REG/n is not a subset of $(\mathrm{REG}/n)_{\mathrm{serial}}$. Let Z be an infinite binary string satisfying (8). Let $L := \{Z \upharpoonright n : n \in \mathbb{N}\}$.

By means of an advice function $h(n) = Z \upharpoonright n$, L is shown to be in REG/n.

We are going to show that L does not belong to $(\mathrm{REG}/n)_{\mathrm{serial}}$. Assume that L belongs to it. Suppose that L'' and g are a regular language and an advice function satisfying the requirements in Definition 3. Suppose that M is a deterministic finite automaton that accepts L''. Let Q be its set of states. For each n, let $q_{(n)}$ be the state when M has read $g(n)$.

We define a deterministic Turing machine N as follows. An input is an ordered pair $(q, n) \in Q \times \mathbb{N}$. For each $y \in \{0,1\}^n$, simulate the moves of M as follows. Set the state (of the virtual M) being q. Let M read y. If M accepts y, return y and halt. If the for-loop finishes without any output, then N does not halt.

By the definition of L'' and g, for the input $(q_{(n)}, n)$, N outputs $Z \upharpoonright n$. In addition, by a certain appropriate coding, we may assume that the domain of N is a prefix free set. For example, code an ordered pair $(u_1 \cdots u_m, v_1 \cdots v_\ell)$ by a string $u_1 u_1 \cdots u_m u_m 01 v_1 v_1 \cdots v_\ell v_\ell 01$.

Therefore, $K_N(Z \upharpoonright n)$ is in the order of the length of $(q_{(n)}, n)$. Thus, it is $O(\log_2(n))$. Hence, by the definition of an optimal machine, $K(Z \upharpoonright n) \leq K_N(Z \upharpoonright n) + O(1) = O(\log_2 n)$. This contradicts to the assumption of (8). □

C. Parallel versus Serial: Realtime Deterministic Pda

If a context-free language is recognized by a cetain deterministic automaton, it is called a deterministic context-free language. The class of all such languages is denoted by DCFL. DCFL is a basic and important subclass of CFL, and has various applications (for example, see [10]).

In this subsection, we review DCFL and its subclass RDCFL. We are going to observe that there exists a CFL-immune set in $\mathrm{RDCFL}(2) - \mathrm{CFL}/n$. In this sense, $\mathrm{RDCFL}(2)$ is a much larger class than CFL.

Main result of this subsection is that RDCFL/n is not a subset of $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$. Finally, we will state some open problems.

Recall that an *ε-move* (*ε-transion*) is a move of a pushdown automaton without advancing the head: In an ε-move, operations on the state and the stack are allowed [8].

Definition 5. Suppose that Σ is an alphabet for input strings and M is a pushdown automaton over Σ.

- [7], [8] M is a *deterministic pushdown automaton* (*dpda*, for short) if for each state q and each stack letter X, the following (i) and (ii) hold: (i) If for some $a \in \Sigma$, a move for (q, a, X) is possible, then M does not perform an ε-move for (q, X). (ii) For each $a \in \Sigma \cup \{\varepsilon\}$, M has at most one possible move for (q, a, X).
- [1] M is a *realtime pushdown automaton* (*rpda*, for short) if M does not have an ε-rule: For any input string, at any move, M advances the head a step further, to the right cell.
- M is a *realtime deterministic pushdown automaton* (*rdpda*, for short) if M is a dpda and rpda.
- We let DCFL (RDCFL, respectively) denote the class of all languages accepted by dpda*s* (rdpda*s*, respectively).

□

Example 4. Languages $L_{2,1}$ and $L_{2,2}$ itroduced in the proof of Lemma 2 are in RDCFL. Therefore, the language L_2 itroduced in Definition 2 is in $\mathrm{RDCFL}(2)$. By Theorem 4, there exists a CFL-immune set in $\mathrm{RDCFL}(2) - \mathrm{CFL}/n$. By Theorem 5, there exists a CFL-immune set in $\mathrm{RDCFL}(2) - (\mathrm{CFL}/n)_{\mathrm{serial}}$. □

Example 5. In Introduction, we observe language $L_{3\mathrm{eq}} = \{0^n 1^n 2^n : n \in \mathbb{N}\}$. This belongs to $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$. In order to verify this, define a serial advice $g(3n)$ as to be $2^n 1^n 0^n$. Consider an rdpda that works as follows. The rdpda pushes the advice into the stack. Then, it pops letters one by one from the stack, and compares them with the letters from the input string. □

Example 5 suggests that separation of RDCFL/n from $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$ requires different approach from Example 1. By extending the method of Example 3, we show the following.

Theorem 6. RDCFL/n *is not a subset of* $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$.

Proof: Let Z be an infinite binary string satisfying (8). Let $L := \{w\ Z \upharpoonright n\ w^R : n \in \mathbb{N}, w \in \{2,3\}^*, |w| = \lceil \sqrt{n} \rceil\}$. In the right-hand side, 2 and 3 are considered to be letters.

By means of an advice function $h(2\lceil \sqrt{n} \rceil + n) = 2^{\lceil \sqrt{n} \rceil} Z \upharpoonright n\ 2^{\lceil \sqrt{n} \rceil}$, L is shown to be in RDCFL/n.

In order to show that L does not belong to $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$, suppose that L belongs to $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$. Take an rdpda M and an advice function g that witness that L is in the serial class.

Suppose that n and m are positive integers such that $m = 2\lceil\sqrt{n}\rceil + n$ and that w is a string over $\{2,3\}$ of length $\lceil\sqrt{n}\rceil$. Suppose that $g(m)\ w\ Z \restriction n\ w^R$ is given as an input of M and that s is an accepting path. Let $g(m)\ w\ Z \restriction n\ w^R = u_1 \cdots u_{2m}$.

By our assumption that M does not perform ε-moves, s is expressed as a sequence $(r^1, s^1), (r^2, s^2), \ldots, (r^{2m}, s^{2m}), (r^{2m+1}, s^{2m+1})$ of the following properties. r_1 is the initial state. $s^1\bot$ is the initial stack string, where s_1 is the empty string, and $\bot$ is the symbol for the bottom of the stack (Fig. 5). For each $i \in \{1, \ldots, 2m\}$, in the state r^i with the stack string $s^i\bot$, M reads the input symbol u_i; then M performs an action, and the resulting state and stack string are r^{i+1} and $s^{i+1}\bot$, respectively (Figs. 6, 7). .

For each i, let $\ell(i)$ denote the length of s^i. For each i such that $\ell(i) > 0$, suppose that $s^i = s^i_{\ell(i)} \cdots s^i_1$. Here, $s^i_{\ell(i)}$ is the top letter of the stack when M is going to perform the action for the head at u_i.

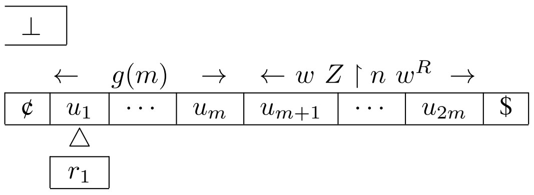


Fig. 5. An rdpda

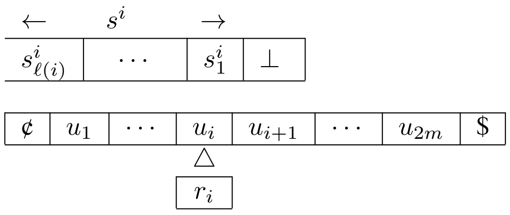


Fig. 6. Before the action for u_i

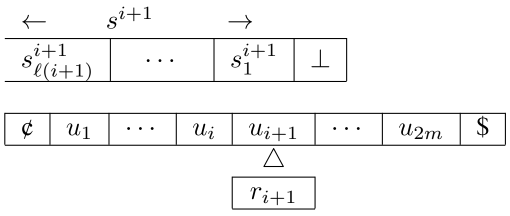


Fig. 7. After the action for u_i

We investigate the interval $m < i \le m + \lceil\sqrt{n}\rceil$, in other words, that during the head is touching letters of w.

Suppose that α is an integer such that the minimum value of $\ell(i)$ in the interval is achieved when $i = \alpha$, in other words, when M is going to perform the action for the head at u^α; in addition, suppose that α is the maximum possible value with this property.

Next, we look at $i = m + \lceil\sqrt{n}\rceil$. In other words, we look at the instance when M is going to perform the action for the head at the last letter of w. Then $s^{m+\lceil\sqrt{n}\rceil}$ is of the form $ts^\alpha\bot$ for some string t.

Claim 1 There exists β such that $m + \lceil\sqrt{n}\rceil < \beta \le m + \lceil\sqrt{n}\rceil + n$ and $s^\beta = s^\alpha$. In other words, during the head is touching letters of $Z \restriction n$, there is an instance when t is completely removed.

Proof of Claim 1: If t is not completely removed during the head is touching letters of $Z \restriction n$, we can find $Z \restriction n$ by using the following: (1) M, (2) r^i for $i = m + \lceil\sqrt{n}\rceil$, that is, the state when M is going to perform the action for the head at the last letter of w, (3) t, (4) the substring s' of s^α consisting of the $\lceil\sqrt{n}\rceil$ letters from the closest to the top, and (5) w^R.

A procedure finiding $Z \restriction n$ is as follows. For each string x of length n, simulate M from the following settiing. The state is r^i of (2); the stack string is $ts'\bot$, where t is the string of (3) and s' is the string of (4); and the input tape is holding xw^R, with the head at the leftmost letter. The virtual M accepts x only in the case when $x = Z \restriction n$. Thus, we can find $Z \restriction n$.

We can produce t by the following: n, M, r^α(the state when M is going to perform the action for u_α), w, and $\alpha - m$ (the relative position of the head in w). These parameters are coded via a binary string of length $O(\sqrt{n})$. Therefore, we can code (1)–(5) in the previous paragraph via a binary string of length $O(\sqrt{n})$.

Then in a way similar to Example 3, the assumption that t is not removed derives a contradiction. Hence, t must be completely removed during the head is touching letters of $Z \restriction n$. Q.E.D.(Claim 1)

Let β be the maximum possible β with the property in Claim 1.

Hence, we can find w^R by n, M, r^β (the state when M is going to perform the action for u_β), $Z \restriction n$, $\beta - \lceil\sqrt{n}\rceil - m$ (the relative position of the head in $Z \restriction n$) and $s^\beta (= s^\alpha)$.

Claim 2 We can find s^α by M, $g(m)$ and $\ell(\alpha)$ (the length of s_α).

Proof of Claim 2: When M has finished to perform the action for the head at the last letter of $g(m)$, the stack string s^{m+1} is of the form $t's^\alpha$ for some string t'. Thus, s^α is the substring of s^{m+1} consisting of $\ell(\alpha)$ letters from the closest to the bottom. Q.E.D.(Claim 2)

Hence, we can find w^R by the following : (i) n, (ii) M, (iii) $Z \restriction n$, (iv) $g(m)$, (v) r^β, (vi) $\beta - \lceil\sqrt{n}\rceil - m$, and (vii) $\ell(\alpha)$.

Under the setting that n, M, Z and g are fixed, w^R is determined by (v)–(vii). With respect to (v), the number of choices is at most the number of states of M. Since (vi) is a relative position in $Z \restriction n$, (vi) is a natural number at most n.

Remind that M does not perform ε-moves. Thus, the length of $t's^\alpha$ $(= s^{m+1}$, see the proof of Claim 2) is $O(m)$, and therefore $O(n)$. Since $\ell(\alpha) \le |t's^\alpha|$, $\ell(\alpha)$ of (vii) is a natural number at most $O(n)$.

Hence, there are at most $O(n^2)$ choices for the values of the parameters determining w^R. However, there are $2^{\lceil\sqrt{n}\rceil}$ choices for w. By the pigeonhole principle, we get a contradiction.

Hence, L does not belong to $(\mathrm{RDCFL}/n)_{\mathrm{serial}}$. ■

To our knowledge, we do not know whether the following hold.

1) $(\mathrm{RDCFL}/n)_{\mathrm{serial}} \subset \mathrm{RDCFL}/n$?
2) $\mathrm{DCFL}/n \subset (\mathrm{DCFL}/n)_{\mathrm{serial}}$?
3) $(\mathrm{DCFL}/n)_{\mathrm{serial}} \subset \mathrm{DCFL}/n$?
4) $\mathrm{CFL}/n \subset (\mathrm{CFL}/n)_{\mathrm{serial}}$?
5) $(\mathrm{CFL}/n)_{\mathrm{serial}} \subset \mathrm{CFL}/n$?

ACKNOWLEDGMENT

We thank Tomoyuki Yamakami, Masahiro Kumabe and Yuki Mizusawa for helpful discussions. We would like to thank the anonymous referee for valuable comments on the earlier version (arXiv:1502.00367v1).

REFERENCES

[1] Autebert, J.-M., Berstel, J. and Boasson, L., "Context-free languages and pushdown automata," In: Rosenberg, G. and Salomaa, A. eds. *Handbook of formal languages: Volume 1. Word, Language, Grammar*, pp.111-174, Springer, 1997.
[2] Bar-Hillel, Y., M. Perles, and E. Shamir, "On formal properties of simple phrase structure grammars," *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, **14** pp.143–172 (1961).
[3] Berstel, J. and L. Boasson, "Context-free languages," In: van Leeuwen, J., eds., *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pp.59–102, Elsevier, 1990.
[4] Damm, C. and M. Holzer, "Automata that take advice," In: *Proc. 20th Symposium on Mathematical Foundations of Computer Sciences*, Lecture Notes in Comput. Sci., **969** pp.149–158, Springer, 1995.
[5] Feige, U. and A. Shamir, "Zero knowledge proofs of knowledge in two rounds," In: *Advances in Cryptography - CRYPTO '89*, Lecture Notes in Comput. Sci., **435** pp.526–544 Springer, 1990.
[6] Flajolet, P. and J.M. Steyaert, "On sets having only hard subsets," In: *Proc. 2nd International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Comput. Sci., **14** pp.446–457, Springer, 1974.
[7] Ginsburg, S. and Greibach, S., "Deterministic context free languages," *Inform. Control*, **9** pp.620–648 (1966).
[8] Hopcroft, J.E. and J.D. Ullman, "*Introduction to Automata Theory, Languages, and Computation*," Addison-Wesley, 1979.
[9] Karp, R.M. and R. Lipton, "Turing machines that take advice," *L'Enseignement Math.*, **28** pp.191–209 (1982).
[10] Nakano, R., "Error correction of enumerative induction of deterministic context-free," L-system grammar. *IAENG International Journal of Computer Science*, **40** pp.47–52 (2013).
[11] Nies, A., "*Computability and Randomness*," Oxford, 2009.
[12] Ogden, W., R.J. Ross and K. Winklmann, "An interchange lemma for context-free languages," *SIAM J. Comput.*, **14** pp.410–415 (1985).
[13] Pippenger, N., "On simultaneous resource bounds," In: *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp.307–311, Springer, 1979.
[14] Shallit, J., "*A Second Course in Formal Languages and Automata Theory*," Cambridge University Press, 2009.
[15] Tadaki, K., T. Yamakami and J.C.H. Lin, "Theory of one-tape linear-time Turing machines," *Theoret. Comput. Sci.*, **411** pp.22–43 (2010).
[16] Yamakami, T., "Swapping lemmas for regular and context-free languages," *preprint*, arXiv:0808.4122v2 (2009). The version 1 is arXiv:0808.4122v1 (2008).
[17] Yamakami, T., "Immunity and pseudorandomness of context-free languages," *Theoret. Comput. Sci.*, **412** pp.6432–6450 (2011).
[18] Yamakami, T. and T. Suzuki, "Resource bounded immunity and simplicity," *Theoret. Comput. Sci.*, **347** pp.90–129 (2005).