Limited Two-Way Deterministic Finite Automata with Advice

Ahmet Bilal Uçan(✉)

Bogazici University, Istanbul, Turkey
ahmet.ucan@boun.edu.tr

Abstract. External assistance in the form of strings called advice is given to an automaton in order to make it a non-uniform model of computation. Automata with advice are then examined to better understand the limitations imposed by uniformity, which is a typical property shared by all feasible computational models. The main contribution of this paper is to introduce and investigate an extension of the model introduced by Küçük et al. [6]. The model is called *circular deterministic finite automaton with advice tape (cdfat)*. In this model the input head is allowed to pass over input multiple times. The number of allowed passes over the input, which is typically a function of input length, is considered as a resource besides the advice amount. The results proved for the model include a hierarchy for cdfat with real-time heads, simulation of 1w/1w cdfat by 1w/rt cdfat, lower bounds of resources provided to a cdfat in order to make it powerful enough to recognize any language, utilizable advice limit regardless of the allowed pass limit, a relation between utilizable pass limit and advice limit, and some closure properties.

Keywords: Formal languages · Automata theory · Advised computation

1 Introduction

Advised computation, where external trusted assistance is provided to a machine to help it for computational tasks, was introduced by Karp and Lipton [4] in 1982. Damm and Holzer [1] considered giving advice to restricted versions of Turing machines. Recent work on finite automata with advice include the papers of Yamakami [8–11], Tadaki et al. [7], Freivalds et al. [3], Küçük et al. [6] and Ďuriš et al. [2]. Today, there are many different models in literature, partly because of the several options available for a machine to access its advice. However, all such models share some common properties. There is an advice function, which maps input lengths to advice strings and not needed to be computable. Advice strings are composed of characters from an advice alphabet. The machine has to use the same advice string when operating on inputs of the same length. We investigate the class of languages recognized by a machine when it consults some advice function having some bounded growing rate. We then play with that upper bound to see what happens to the aforementioned class. An advised automaton takes advantage of an advice string by reading the character under the advice head and choosing appropriate transition from its transition function accordingly. So the same machine may recognize different languages using different advice functions.

We focus on the advice tape model introduced by Küçük et al. in [6]. Since that model becomes extremely powerful (able to recognize all languages) when allowed to use a 2-way input head, and is remarkably limited for the 1-way head case, [2, Theorem 2], [6, Theorem 13], we examine a limited version of two-way input access.

Some common terminology to be used in this paper are as follows: n denotes input length, M denotes an automaton, L denotes a language, h denotes an advice function, w denotes a string, Σ denotes input alphabet, Γ denotes advice alphabet, $\star$ means any, ALL denotes the set of all languages, and $|w|_c$ denotes the number of occurrences of character c in string w.

Here are some definitions of concepts that will be used in our discussion,

Definition 1 [6]. $w_1 \equiv_{L,n,k} w_2 \iff w_1, w_2 \in \Sigma^k \wedge \forall z \in \Sigma^{n-k}[w_1 z \in L \iff w_2 z \in L]$.

Definition 2 [2, Definition 5]. *Let $\{R_n\}_{n=1}^{\infty}$ be a family of relations $R_n \subseteq \Sigma^n \times \Sigma^{f(n)}$ for some $f : \mathbb{N} \to \mathbb{N}$ such that $\forall x_0, x_1 \in \Sigma^n, x_0 \neq x_1$, there is a $y \in \Sigma^{f(n)}$ such that $R_n(x_i, y)$ and $\neg R_n(x_{1-i}, y)$ for some $i \in \{0, 1\}$. Let L_R be the language $L_R := \{xy | x \in \Sigma^\star, |y| = f(|x|), R_{|x|}(x, y)\}$. We call L_R a prefix-sensitive language for relation family R.*

Definition 3. *We call L a prefix-sensitive language iff there exists a relation family R such that L_R is a prefix-sensitive language for relation family R.*

2 Our Model

We defined this model and decided to work on it because the model seems to provide a smooth passage from one-way input head to two-way input head. The name of the new model is *circular deterministic finite automaton with advice tape* (cdfat) which may have real-time or 1-way input and advice heads (4 possible versions). Circular machines read their input circularly, that is, when the input endmarker has seen and the next transition dictates machine to move its input head to right, the input head immediately returns to the beginning position. Advice head is not allowed to perform such a move.

Note that when restricted to a single pass on input, this model is exactly the same with the standard *deterministic finite automaton with advice tapes* model (except the two-way input head version) introduced by Küçük et al. [6].

2.1 Definition

A circular deterministic finite automaton is a 9-tuple $(Q, \Sigma, \Gamma, T_I, T_A, \delta, q_0, q_{acc}, q_{rej})$ where

(i) Q is a finite set of internal states,
(ii) Σ is a finite set of symbols called the input alphabet that does not contain the endmarker symbol, \$, such that $\$ \notin \Sigma$ and $\Sigma' = \Sigma \cup \{\$\}$,
(iii) Γ is a finite set of symbols called advice alphabet that does not contain the endmarker symbol, \$, such that $\$ \notin \Gamma$ and $\Gamma' = \Gamma \cup \{\$\}$,
(iv) $T_I \in \{\{S, R\}, \{R\}\}$ represents the set of allowed input head movements where S and R means *stay-put* and *right* respectively,
(v) $T_A \in \{\{S, R\}, \{R\}\}$ represents the set of allowed advice head movements where S and R means *stay-put* and *right* respectively,
(vi) $q_0 \in Q$ is the initial state on which the execution begins,
(vii) $q_{acc} \in Q$ is the accept state on which the execution halts and accepts,
(viii) $q_{rej} \in Q$ is the reject state on which the execution halts and rejects,
(ix) $\delta : Q \times \Sigma \times \Gamma \to Q \times T_I \times T_A$ is the transition function such that, $\delta(q_1, \sigma, \gamma) = (q_2, t_I, t_A)$ implies that when the automaton is in state $q_1 \in Q$ and it scans $\sigma \in \Sigma'$ on its input tape and $\gamma \in \Gamma'$ on its advice tape, a transition occurs which changes the state of the automaton to $q_2 \in Q$, meanwhile moving the input and advice tape heads in the directions specified respectively by $t_I \in T_I$ and $t_A \in T_A$,

A cdfat $M = (Q, \Sigma, \Gamma, T_I, T_A, \delta, q_0, q_{acc}, q_{rej})$ is said to accept (reject) a string $x \in \Sigma^*$ with the help of an advice string $a \in \Gamma^*$ if and only if M, when started at its initial state q_0 with $x\$$ on the input tape and $a\$$ on the advice tape and while the tape heads scan the first symbols, reaches the accepting (rejecting) state, q_{acc} (q_{rej}), by changing states and moving the input and advice tape heads as specified by its transition function, δ.

A language L defined on the alphabet Σ, is said to be recognized by such a cdfat M with the help of an advice function $h : \mathbb{N} \to \Gamma^*$ if and only if

- $L = \{x \mid M$ accepts x with the help of $h(|x|)\}$, and
- $\bar{L} = \{x \mid M$ rejects x with the help of $h(|x|)\}$.

A language L is said to be recognized by a cdfat, M, using $O(g(n))$-length advice if there exists an advice function h with the following properties:

- $|h(n)| \in O(g(n))$, and
- M recognizes L with the help of $h(n)$.

A language L is said to be recognized by a cdfat, M, using $f(n)$ passes over the input if and only if during the execution of any input of length n, transitions of the form $\delta(\_, \$, \_) = (\_, R, \_)$ are used at most $f(n)$ times in total.

Note that it is not allowed for a cdfat to have a transition of the form $\delta(\_, \_, \$) = (\_, \_, R)$, however, there can be transitions $\delta(\_, \$, \_) = (\_, R, \_)$. The endmarker of the input is for informing the machine. It may be a different model if we omit it, for the sake of backward compatibility we continue to use it.

For the notational purposes, $\mathcal{L}\{rt - f(n)\}$ denotes the set of languages recognized by cdfat with real-time heads, $(n+1)f(n)$ length advice and $f(n)$ passes. When a head is allowed to stay-put on its tape, we use a different notation. For instance $\mathcal{L}\{1 - [f(n)]/g(n)\}$ denotes the set of languages recognized by cdfat with 1-way input head and real-time advice head, using $g(n)$ length advice and $f(n)$ passes.

2.2 Results

Theorem 1. *A language L is prefix-sensitive if and only if for all $k \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that $\equiv_{L,n,k}$ has $|\Sigma|^k$ equivalence classes.*

Proof. Assume that for some language L it holds that for all $k \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that $\equiv_{L,n,k}$ has $|\Sigma|^k$ equivalence classes. Let f be a function which maps any $k \in \mathbb{N}$ to an n so that $\equiv_{L,n,k}$ has $|\Sigma|^k$ equivalence classes. Define an infinite family of relations $\{R_k\}_{k=1}^{\infty}$ such that $R_k \subseteq \Sigma^k \times \Sigma^{f(k)-k}$ and for all $x \in \Sigma^k$ and all $y \in \Sigma^{f(k)-k}$, $xy \in L \iff R_k(x,y)$. It holds that $\forall x_0, x_1 \in \Sigma^k, x_0 \neq x_1$, there is a $y \in \Sigma^{f(k)-k}$ such that $R_k(x_i, y)$ and $\neg R_k(x_{1-i}, y)$ for some $i \in \{0,1\}$. Because if there were no such y for some x_0 and x_1, then $x_0 \equiv_{L,f(k),k} x_1$ would be true and the number of equivalence classes would not be $|\Sigma|^k$. According to the Definition 2, we concluded that L is prefix-sensitive.

For the other direction, let L be a prefix-sensitive language. According to the Definition 2, $L = \{xy | x \in \Sigma^\star, R_{|x|}(x,y)\}$ where $f : \mathbb{N} \to \mathbb{N}$ is a function and $\{R_k\}_{k=1}^{\infty}$ is an infinite sequence of relations such that $R_k \subseteq \Sigma^k \times \Sigma^{f(k)}$ and $\forall x_0, x_1 \in \Sigma^k, x_0 \neq x_1$, there is a $y \in \Sigma^{f(k)}$ such that $R_k(x_i, y)$ and $\neg R_k(x_{1-i}, y)$ for some $i \in \{0,1\}$. It holds that for all $k \in \mathbb{N}$, $\equiv_{L,k+f(k),k}$ has $|\Sigma|^k$ equivalence classes. Because if the number of equivalence classes of $\equiv_{L,k+f(k),k}$ is less than $|\Sigma|^k$ for some k, then there would be two strings x_0 and x_1 such that $x_0 \equiv_{L,k+f(k),k} x_1$ and that would imply that there is no y of length $f(k)$ such that $R_k(x_i, y)$ and $\neg R_k(x_{1-i}, y)$ for some $i \in \{0,1\}$. □

Theorem 2. $\mathcal{L}\{rt - 2^{O(n)}\} = ALL$.

Proof. Let $h(n) = w_1 c_1 w_2 c_2 \ldots w_{|\Sigma|^n} c_{\Sigma^n}$ where each w_i is a distinct input word of length n and each $c_i \notin \Sigma$ is either the accept or the reject symbol. Devise a machine M such that, it tries to match the input word and advice character by character in real-time execution. If a mismatch occurs while trying to match the input word, machine M will advance its input head until it is at the beginning position again. Note that the advice head will be at the first character of the next word on advice at the end of this process. Then it tries to match the next word and so on. At some point matching ends with success, that is, machine M will see the endmarker of input while trying to match the characters. At that point it will accept or reject the string depending on which c_i character it is seeing on the advice. □

Theorem 3. *For any function $f : \mathbb{N} \to \mathbb{N}$, $\mathcal{L}\{rt - f(n)\} = \mathcal{L}\{rt - O(f(n))\}$.*

Proof. The idea is that for any given machine M, one can devise a new machine M' such that M' uses k times less passes than M for all n and for an arbitrary $k \in \mathbb{N}$, and still recognizes the same language with the help of some other advice function. Let us group the passes of machine M so that i^{th} group consists of passes from $(i-1)k+1$ to ik. With a single pass, machine M' simulates a group of k passes of M. First pass simulates the first group and second pass simulates the second group and so on. Since M' does not know which state to begin with a pass without knowing the result of the previous one, it simulates all possibilities and remembers the final state of the previous group of passes using again its states. Therefore the size of the state set of M' is s^{ks+1} where s is the number of states of M.

The new advice function h' is a compressed version of the old one. Let Γ' be the new advice alphabet whose symbols represent the k permutations of the symbols of Γ. $|\Gamma'| = |\Gamma|^k$ holds. Let $|h'(n)| = |h(n)|/k$ for all $n > 0$. Note that without loss of generality we assume that $|h(n)|$ is an integer multiple of k. We prepare the new advice strings so that $h'(1)$ represents all strings from $h(1)$ to $h(k)$, $h'(2)$ represents all strings from $h(k+1)$ to $h(2k)$ and so on. □

Theorem 4. *For any function $f : \mathbb{N} \to \mathbb{N}$, $L_1, L_2 \in \mathcal{L}\{rt - f(n)\} \implies L_1L_2 \in \mathcal{L}\{rt - nf(n)\}$.*

Proof. Let M_1, M_2 be machines recognizing L_1, L_2 with the help of advice functions h_1, h_2 respectively. Let M_3 be the machine which is claimed to recognize the concatenation language L_1L_2 with the help of advice function h_3. The idea is to predict the words $w_1 \in L_1$ and $w_2 \in L_2$ such that w_1w_2 is the input word. Machine M_3 doesn't know from where to divide the input, so it just tries all the possibilities. We replace the advice characters whose locations correspond to the last character of the first portion of the input with their marked versions in order to inform the machine M_3.

In the first pass over the input, machine M_3 first simulates M_1 on the first portion of the input and stores the last internal state of that execution. Then it simulates M_2 on the rest of the input and stores the last state of that execution too. Then it begins the second pass simulating M_1 again but this time starting from the last saved state of that thread and when it completes, M_3 will update the last state of the thread and so on. Throughout the execution of M_3, two separate threads of execution are simulated at the same time. At the end of at most $f(n)$ passes, if both threads end with accepting their respective sub-inputs, M_3 accepts the input. Otherwise, M_3 continues the computation with a different division of the input. Note that, given an input word of length n, there are $n+1$ different pairs of words such that their concatenation is the input word. At the end of at most $(n+1)f(n)$ passes, if no division works, M_3 rejects the input. According to the Theorem 3, asymptotic rate of the passes is the important part so $nf(n)$ passes can do the same job.

Note that we should double the old advice alphabet size and introduce marked versions of the old symbols in order to mark the position of input separation on the advice h_3. Also note that advice string $h_3(n)$ will be an interleaved version of the $h_1(k)$ and $h_2(n-k)$ concatenated for all $k \in [0, n]_{\mathbb{Z}}$. □

Corollary 1. *$\mathcal{L}\{rt - poly\}$ is closed under concatenation.*

Lemma 1. *Let $L \in \mathcal{L}\{rt - f(n)\}$. Then for all n and for all k smaller than n, $\equiv_{L,n,k}$ has $2^{O(f(n))}$ equivalence classes.*

Proof. Let $s = |Q|$ and let $Q' = \{q_1, q_2, \dots q_{s-2}\}$ be the set of all states except the accept and reject states. Let $\alpha_{w,i} : Q' \to Q'$ be a mapping which maps the internal state of machine when input head is for the first time on the first character of w and advice head is at the i^{th} position to the internal state of machine when input head is for the first time on the first character right after the w. Besides its parameters w and i, this mapping depends on the content of the advice and transition function of the machine. Here we consider a single machine working on inputs of the same length n therefore the mapping depends only on its parameters.

Consider execution of a real-time circular machine on two different inputs of length n, namely w_1z and w_2z. If we can find two words w_1 and w_2 such that $\alpha_{w_1,i} = \alpha_{w_2,i}$ for all $i \in \{1, 1+(n+1), \dots 1+(f(n)-1)(n+1)\}$ then the two inputs must have the same fate for all z.

Given a single i, there are less than s^s distinct functions $\alpha_{w,i}$. Considering all $f(n)$ functions mentioned above for a word w, there are less than $(s^s)^{f(n)}$ different permutations. Assuming that the number of equivalence classes of relation $\equiv_{L,n,k}$ is greater than $(s^s)^{f(n)}$ for some k and n, there would be two words w_1 and w_2 such that they are in different equivalence classes and have all the same mappings. This is a contradiction. □

Theorem 5. *Let $f(n), g(n) \in O(n)$ and $f(n) \in o(g(n))$. Then $\mathcal{L}\{rt - f(n)\} \subsetneq \mathcal{L}\{rt - g(n)\}$.*

Proof. Consider the language family $L_\rho = \{w^{\rho(|w|)} |\ w \in \Sigma^*,\ \rho : \mathbb{N} \to \mathbb{N}\}$. Note that ρ is assumed to be a non-decreasing function and the input length $n = |w|\rho(|w|)$. Inputs consist of repetitions of a substring w. Define $\phi(m\rho(m)) = m$ for all $m \in \mathbb{N}^+$. Depending on the choice of ρ, $\phi(n) \in \omega(1) \cap O(n)$. We will give three lemmas. Two of them show a hierarchy for the range $\omega(1) \cap O(n)$ and the last one is to put the $\Theta(1)$ in.

Lemma 2. $L_\rho \in \mathcal{L}\{rt - \phi(n)\}$.

Since given the input length n and the function ρ we can deduct the period of input, we can check a position of the repeating substring w for each pass. Therefore our machine will need $|w| = \phi(n)$ many passes.

The advice strings are of the form (parentheses are meta-characters),

$$h(n) = (10^{|w|-1})^{\rho(|w|)}\#(010^{|w|-2})^{\rho(|w|)} \dots \#(0^{|w|-1}1)^{\rho(|w|)}$$

Our machine will first search for the first 1 on advice tape and when it has been found, the machine saves the corresponding input character in its states and continue searching for the next 1. When it sees the next 1 it checks the corresponding input character with the one it saved before. If they mismatch input is rejected. The machine then continue searching for 1s and do the same checking till the end of the first pass. It then start with the second pass and do the same procedure again, checking the equality of next character position in substring w. If the endmarker of advice is reached, input is accepted.

Lemma 3. *$L_\rho \notin \mathcal{L}\{rt - o(\phi(n))\}$.*

Observe that any L_ρ is prefix-sensitive. Thinking each word as concatenation of first period w and the rest, in other words selecting k to be $\phi(n)$ for all n, $\equiv_{L_\rho,n,k}$ has $|\Sigma|^{\phi(n)}$ equivalence classes. According to the Lemma 1, $L_\rho \notin \mathcal{L}\{rt - o(\phi(n))\}$.

Lemma 4. *No prefix-sensitive language is in $\mathcal{L}\{rt - O(1)\}$.*

According to the Lemma 1, for any language $L \in \mathcal{L}\{rt - O(1)\}$, $\equiv_{L,n,k}$ has $2^{O(1)} = O(1)$ equivalence classes. Therefore according to Theorem 1, L is not prefix sensitive. □

Theorem 6. *Let $L_1 \in \mathcal{L}\{1 - [f(n)]/1 - [g(n)]\}$ and $L_2 \in \mathcal{L}\{1 - [f'(n)]/1 - [g'(n)]\}$. Then $L_1 \cup L_2 \in \mathcal{L}\{1 - [f(n) + f'(n)]/1 - [g(n) + g'(n)]\}$.*

Proof. Let M_1, M_2 be machines recognizing languages L_1, L_2 with the help of advice functions h_1 and h_2 respectively. Devise a new advice function,

$$h_3(n) = h_1(n)\#h_2(n)$$

for all n where # is a brand new advice character that occurs nowhere else. Let M_3 be the machine recognizing the union language with the help of h_3. Machine M_3 first simulates the M_1 and during this simulation it treats the # character in advice as an endmarker. When this simulation ends, which may take at most $f(n)$ passes over the input, M_3 stores the result in its states and start simulating M_2 after adjusting its heads to proper positions, that is input head to the beginning and advice head to the next character after #. After at most $f'(n)$ passes over the input, it completes the execution and store the result in its states. In this way it may end up in 4 different states for 4 possible acceptance status of M_1 and M_2. Via combining some of those states into the accept state and the rest into the reject state; union, intersection or difference of L_1 and L_2 are all recognizable. □

Corollary 2. *$\mathcal{L}\{1 - [O(f(n))]/1 - [O(g(n))]\}$ is closed under union and intersection.*

Theorem 7. *For any function $f : \mathbb{N} \to \mathbb{N}$, $\mathcal{L}\{1 - [f(n)]/1 - [\star]\} = \mathcal{L}\{1 - [f(n)]/1 - [2^{O(n)}]\}$.*

Proof. The proof is an easy modification of the proof given by Ďuriš et al. for [2, Theorem 3]. □

Theorem 8. *For any function $g : \mathbb{N} \to \mathbb{N}$, $\mathcal{L}\{1 - [\star]/1 - [g(n)]\} = \mathcal{L}\{1 - [O(g(n))]/1 - [g(n)]\}$.*

Proof. Consider the execution of an s-state cdfat with one-way heads. Pausing the advice head, passing on the input more than s times forces the machine to enter an infinite loop. Thus, a machine must advance its advice head before that threshold. Therefore at most $sg(n)$ passes are possible for an execution which eventually halts. □

Theorem 9. *For any functions $f, g : \mathbb{N} \to \mathbb{N}$, $\mathcal{L}\{1-[f(n)]/1-[g(n)]\} \subseteq \mathcal{L}\{1-[f(n)]/O(nf(n)g(n))\}$.*

Proof. It is possible to simulate the one-way advice head with real-time advice head using additional advice. The idea is to replicate each advice character $(n+1)f(n)$ times and use separator characters # to mark the transition locations. That is, for all n,

$$h(n) = c_1c_2\ldots c_k \implies h'(n) = c_1^{(n+1)f(n)}\#c_2^{(n+1)f(n)}\#\ldots c_k^{(n+1)f(n)}\#c_\$^{(n+1)f(n)}$$

where $c_i \in \Gamma$ for all $i \in \{1, 2\ldots k\}$ and the $c_\$$ is a new advice character which is for repeating the endmarker (it is not allowed to have more than one real endmarker character). When the new machine reads $c_\$$ on h', it behaves exactly like the old machine seeing endmarker on h.

Instead of stay-putting advice head in old machine, let it move right one step in new machine. Instead of moving advice head one step in old machine, enter a subprogram which takes advice head to the next # character in new machine.

This trick works because a cdfat with one-way heads must forward its advice head within $(n+1)f(n)$ computational steps. This is because without loss of generality we can assume at least one head is moving in each step and of course input head can move at most $(n+1)f(n)$ times in an execution. □

Corollary 3. *For any function $f : \mathbb{N} \to \mathbb{N}$, $\mathcal{L}\{1-[f(n)]/1-[poly]\} = \mathcal{L}\{1-[f(n)]/poly\}$.*

It is already known that dfat with 2-way input head is equal in power with the prefix advice model when provided with constant advice [5, Theorem 3.8]. Since our model is sandwiched in between the 2-way input model and advice prefix model when it comes to power, we deduce that $\mathcal{L}\{1-[i]/1-[k]\} = \mathcal{L}\{1-[i+1]/1-[k]\}$ for all $i \in \mathbb{N}$. Therefore an interesting question to ask is what is the minimum advice for which more passes over input enlarges the class of languages recognized. Küçük and others showed that when provided with polynomial advice, 2-way input head is more powerful than 1-way head [6, Theorem 14]. We proved a stronger result and gave an ultimate answer to the aforementioned question. It turns out that even 2 passes over input is more powerful than a single pass when the machine is provided with an increasing advice.

Theorem 10. *Let $f : \mathbb{N} \to \mathbb{N}$ be any function in $\omega(1)$. Then $\mathcal{L}\{1-[1]/1-[f(n)]\} \subsetneq \mathcal{L}\{1-[2]/1-[f(n)]\}$.*

Proof. Consider the language family $L_\rho = \{w|w \in \{1,2,3\}^*, |w|_1 = |w|_2 = \rho(|w|)\}$. The following two lemmas establish the proof.

Lemma 5. $L_\rho \notin \mathcal{L}\{1-[1]/1-[O(\rho(n))]\}$.

Proof. Küçük et al. proved that for any advice length function f, if $L \in \mathcal{L}\{1-[1]/1-[f(n)]\}$, then for all n and all $k \le n$, $\equiv_{L,n,k}$ has $O(f(n))$ equivalence classes, [6, Lemma 6]. It can be shown that for all n, there exists $k \leq n$ such that $\equiv_{L_\rho,n,k}$ has $\Theta(\rho^2(n))$ equivalence classes. Since $\rho(n) \in \omega(1) \implies \rho(n) \in o(\rho^2(n))$, we conclude that $L_\rho \notin \mathcal{L}\{1-[1]/1-[O(\rho(n))]\}$.

Lemma 6. $L_\rho \in \mathcal{L}\{1-[2]/1-[O(\rho(n))]\}$.

Proof. The idea is to devise a machine which in first pass counts the character 1 and in second pass counts the character 2. Let $L_1 = \{w | w \in \{1,2,3\}^*, |w|_1 = \rho(|w|)\}$ and $L_2 = \{w | w \in \{1,2,3\}^*, |w|_2 = \rho(|w|)\}$. Observe that L_1 or L_2 can easily be recognized by a cdfat with a single pass. In order to recognize L_1 for instance, let $h(n) = 1^{\rho(n)}$ be the advice function, then consider a machine which stay-puts its advice head when it sees a character other than 1 on its input and advances its advice head when it sees 1 on input. It will accept a string iff both endmarkers are read at the same time. L_2 can be recognized similarly. Since $L_1, L_2 \in \mathcal{L}\{1-[1]/1-[O(\rho(n))]\}$, according to Theorem 6, $L_1 \cap L_2 = L_\rho \in \mathcal{L}\{1-[2]/1-[O(\rho(n))]\}$. □

Lemma 7. *Let $L \in \mathcal{L}\{1-[f(n)]/1-[g(n)]\}$. Then for all n and for all k smaller than n, $\equiv_{L,n,k}$ has $2^{O(g(n)\log g(n))}$ equivalence classes.*

Proof. Define a configuration c_i of a machine to be the pair of internal state and advice position. Define a non-stopping configuration $c = (q, m)$ of a machine to be any configuration where q is a state other than accept and reject states. Let $C = \{c_1, c_2, \ldots, c_{(s-2)(g(n)+1)}\}$ be the set of all non-stopping configurations for a machine and for input length n ($s = |Q|$). Without loss of generality assume our machines always end their execution when input head is on the endmarker. Let w be a substring of input (not containing the endmarker) and let $\alpha_w : C \to C$ be a mapping which maps the configuration of machine when the first character of word w is read first time on input tape to the configuration of machine when the character right after the word w is read first time on input tape. Function α depends on transition function of the machine, the specific word w being processed and the advice content. We focus on a single machine and inputs of the same length n, therefore in our case α depends only on w.

Consider execution of a circular machine on two different inputs of length n, namely $w_1 z$ and $w_2 z$. Both inputs start execution at the initial configuration and after each pass they start with a new configuration. If we can find two words w_1 and w_2 such that $\alpha_{w_1} = \alpha_{w_2}$ then the two inputs $w_1 z$ and $w_2 z$ must have the same fate for all z.

There are less than $2sg(n)^{2sg(n)}$ distinct functions α. Assuming that the number of equivalence classes of $\equiv_{L,n,k}$ is greater than $2sg(n)^{2sg(n)}$ for some k and n, there would be two words w_1, w_2 in two different equivalence classes such that they have the same mapping. This is a contradiction. □

Theorem 11. *Let $f(n) \in \omega(1) \cap o(\log n)$. Then the classes $\mathcal{L}\{rt - f(n)\}$ and $\mathcal{L}\{1-[1]/1-[\star]\}$ are incomparable.*

Proof. Recall that $\mathcal{L}\{1-[1]/1-[\star]\}$ is nothing but our way of notating the class of languages recognized by the model introduced by Küçük et al. in [6] given access to unlimited advice. According to Theorem 5, a prefix-sensitive language is in $\mathcal{L}\{rt - f(n)\}$ no matter how slow $f(n)$ grows. However we know from Ďuriš et al. [2, Theorem 2] that no prefix-sensitive language is in $\mathcal{L}\{1 - [1]/1 - [\star]\}$. Therefore $\mathcal{L}\{rt - f(n)\} \nsubseteq \mathcal{L}\{1 - [1]/1 - [\star]\}$.

On the other hand, as stated in proof of Lemma 6, the language $L = \{w | w \in \{1, 2, 3\}^*, |w|_1 = \rho(|w|)\}$ can be easily recognized by a machine with one-way heads, given access to $\Theta(\rho(n))$ length advice. It is easy to see that for all n, there exists k such that $\equiv_{L_\rho, n, k}$ has $\Theta(\rho(n))$ equivalence classes. When the $\rho(n)$ is selected to be linear in n, according to Lemma 1, $L \notin \mathcal{L}\{rt - f(n)\}$. Therefore $\mathcal{L}\{1 - [1]/1 - [\star]\} \nsubseteq \mathcal{L}\{rt - f(n)\}$. $\square$

An interesting question to ask is what is the minimum advice or pass needed in order for a model to recognize any language. We can show some lower bounds using Lemmas 1 and 7. PAL is the language of even palindromes.

Corollary 4. $g(n) \log g(n) \in o(n) \implies PAL \notin \mathcal{L}\{1 - [f(n)]/1 - [g(n)]\}$.

Corollary 5. $f(n) \in o(n) \implies PAL \notin \mathcal{L}\{rt - f(n)\}$.

3 Conclusions and Open Questions

We showed that cdfat with real-time heads can utilize up to linearly many passes over input. We showed that with exponential pass, the real-time machine can recognize any language. However we do not know if the machine can utilize more than linear passes. There may be a clever algorithm for recognizing any language with linear passes.

We showed that even the most powerful version of the cdfat, that is the one having one-way input and advice heads, cannot recognize some languages when there is not enough advice (a nearly linear bound). However we are not aware of an algorithm for this machine which uses less than exponential resources to recognize any language. It would be nice to know the minimum amount of resources needed to recognize any language.

We compared the class of languages recognized by single pass deterministic finite automaton with one-way heads and unlimited advice with the growing class of languages recognized by a real-time cdfat as we allow more passes over input. Since we know that the former class is bigger than the latter when we allow only constant amount of pass over input and the reverse is true when we allow exponential passes over input, we wonder how that growing takes place and is there any pass limit for which the two classes are equal. It turned out that this is not the case. As long as the allowed pass limit is not constant and sub-logarithmic, two classes are not subsets of each other. However we do not know exactly when the latter class encompasses the former one.

Acknowledgements. Thanks to Prof. Cem Say who have helped editing the paper and correcting my mistakes and to my family for their constant support and love.

References

1. Damm, C., Holzer, M.: Automata that take advice. In: Wiedermann, J., Hájek, P. (eds.) MFCS 1995. LNCS, vol. 969, pp. 149–158. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60246-1_121
2. Ďuriš, P., Korbaš, R., Královič, R., Královič, R.: Determinism and nondeterminism in finite automata with advice. In: Böckenhauer, H.-J., Komm, D., Unger, W. (eds.) Adventures Between Lower Bounds and Higher Altitudes. LNCS, vol. 11011, pp. 3–16. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98355-4_1
3. Freivalds, R.: Amount of nonconstructivity in deterministic finite automata. Theoret. Comput. Sci. **411**(38–39), 3436–3443 (2010). https://doi.org/10.1016/j.tcs.2010.05.038
4. Karp, R., Lipton, R.: Turing machines that take advice. Enseign. Math. **28**, 191–209 (1982)
5. Küçük, U.: Finite and small-space automata with advice. Ph.D. thesis, Boğaziçi University (2018)
6. Küçük, U., Say, A.C.C., Yakaryılmaz, A.: Finite automata with advice tapes. In: Béal, M.-P., Carton, O. (eds.) DLT 2013. LNCS, vol. 7907, pp. 301–312. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38771-5_27
7. Tadaki, K., Yamakami, T., Lin, J.C.H.: Theory of one-tape linear-time turing machines. Theoret. Comput. Sci. **411**(1), 22–43 (2010). https://doi.org/10.1016/j.tcs.2009.08.031
8. Yamakami, T.: Swapping lemmas for regular and context-free languages with advice. CoRR abs/0808.4122 (2008). http://arxiv.org/abs/0808.4122
9. Yamakami, T.: The roles of advice to one-tape linear-time turing machines and finite automata. Int. J. Found. Comput. Sci. **21**(6), 941–962 (2010). https://doi.org/10.1142/S0129054110007659
10. Yamakami, T.: Immunity and pseudorandomness of context-free languages. Theoret. Comput. Sci. **412**(45), 6432–6450 (2011). https://doi.org/10.1016/j.tcs.2011.07.013
11. Yamakami, T.: One-way reversible and quantum finite automata with advice. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 526–537. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28332-1_45