Behavioral Strengths and Weaknesses of Various Models of Limited Automata

Tomoyuki Yamakami(✉)

Faculty of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan
TomoyukiYamakami@gmail.com

Abstract. We examine the behaviors of various models of k-limited automata, which naturally extend Hibbard's [Inf. Control, vol. 11, pp. 196–238, 1967] scan limited automata, each of which is a linear-bounded automaton satisfying the k-limitedness requirement that the content of each tape cell should be modified only during the first k visits of a tape head. One central model is k-limited probabilistic automaton (k-lpa), which accepts an input exactly when its accepting states are reachable from its initial state with probability more than 1/2. We further study the behaviors of one-sided-error and bounded-error variants of such k-lpa's as well as deterministic and nondeterministic models. We discuss fundamental properties of those machine models and obtain inclusions and separations among language families induced by these machine models. In due course, we study special features—the blank skipping property and the closure under reversal—which are keys to the robustness of k-lpa's.

Keywords: Limited automata · Pushdown automata
Probabilistic computation · Bounded-error probability
One-sided error · Blank skipping property · Reversal

1 Background and Main Contributions

1.1 Limited Automata and Probabilistic Computation

In 1967, Hibbard [3] studied a novel computational model of so-called *scan limited automata* to characterize context-free languages by conducting direct simulations between one-way nondeterministic pushdown automata (or 1npda's) and his model. Hibbard's model seems to have been paid little attention until Pighizzini and Pisoni [10] reformulated the model from a modern-machinery perspective and reproved a characterization theorem of Hibbard in a more sophisticated manner. A *k-limited automaton*,[1] for each fixed index $k \geq 0$, is in general a one-tape (or a single-tape) Turing machine whose tape head is allowed

[1] Hibbard's original formulation of "k-limited automaton" is equipped with a semi-infinite tape that stretches only to the right with no endmarker but is filled with the blank symbols outside of an input string. Our definition in this paper is different from Hibbard's but it is rather similar to Pighizzini and Pisoni's [10].

© Springer Nature Switzerland AG 2019
B. Catania et al. (Eds.): SOFSEM 2019, LNCS 11376, pp. 519–530, 2019.
https://doi.org/10.1007/978-3-030-10801-4_40

to rewrite each tape cell between two endmarkers only during the first k scans or visits (except that, whenever a tape head makes a "turn," we count this move as double visits). Although these automata can be viewed as a special case of linear-bounded finite automata, the restriction on the number of times that they rewrite tape symbols brings in quite distinctive effects on the computational power of the underlying automata, different from other restrictions, such as upper bounds on the numbers of nondeterministic choices or the number of tape-head turns. Hibbard actually proved that *k-limited nondeterministic automata* (or k-lna's) for $k \geq 2$ are exactly as powerful as 1npda's, whereas 1-lna's are equivalent in power to 2-way deterministic finite automata (or 2dfa's) [12].

In a subsequent paper [11], Pighizzini and Pisoni discussed a close relationship between *k-limited deterministic automata* (or k-lda's) and one-way deterministic pushdown automata (or 1dpda's). In fact, they proved that 2-lda's embody exactly the power of 1dpda's; in contrast, Hibbard observed that, when $k \geq 3$, k-lda's do not, in general, coincide in computational power with 1dpda's. This observation gives a clear structural difference between determinism and nondeterminism on the machine model of "limited automata" and this difference naturally raises a question of whether other variants of limited automata matches their corresponding models of one-way pushdown automata.

Lately, a computation model of *one-way probabilistic pushdown automata* (or 1ppda's) has been discussed extensively to demonstrate computational strengths as well as weaknesses in [5,7,9,17]. Hromkovič and Schnitger [5] as well as Yamakami [17], in particular, demonstrated clear differences in computational power between two pushdown models, 1npda's and 1ppda's.

While nondeterministic computation is purely a theoretical notion, probabilistic computation could be implemented in real life by installing a mechanism of generating (or sampling) random bits (e.g., by flipping fair or biased coins). A *bounded-error* probabilistic machine makes error probability bounded away from $1/2$, whereas an *unbounded-error* probabilistic machine allows error to take arbitrarily close to probability $1/2$. In most cases, a probabilistic approach helps us solve a target mathematical problem algorithmically faster, and probabilistic (or randomized) computation often exhibits its superiority over its deterministic counterpart. For example, 2-way probabilistic finite automata (or 2pfa's) running in expected exponential time can recognize non-regular languages with bounded-error probability [2]. By contrast, when restricted to expected subexponential runtime, bounded-error 2pfa's recognize only regular languages [1,6]. As this example shows, the expected runtime bounds of probabilistic machines largely affect the computational power of the machines, and thus its probabilistic behaviors significantly differ from deterministic behaviors.

The usefulness of probabilistic algorithms motivates us to take a probabilistic approach toward an extension of Hibbard's original model of k-limited automata. This paper in fact introduces *k-limited probabilistic automata* (or k-lpa's) and their variants, including one-sided-error and bounded-error variants, and to explore their fundamental properties to obtain strengths and weaknesses of families of languages recognized by those machine models.

1.2 Main Contributions

Our first goal is to provide in the field of probabilistic computation a complete characterization of finite and pushdown automata in terms of limited automata. All probabilistic machines are assumed to run in expected polynomial time.

For any error bound $\varepsilon \in [0, 1/2)$, the notations $1\mathrm{PPDA}_\varepsilon$ and $2\mathrm{PFA}_\varepsilon$ refer to the families of all languages recognized by ε-error 1ppda's and ε-error 2pfa's, respectively. As a restriction of $2\mathrm{PFA}_\varepsilon$, $2\mathrm{RFA}_\varepsilon$ denotes the family of all languages recognized by 2pfa's with one-sided error probability at most ε. Similarly, we define $1\mathrm{RPDA}_\varepsilon$ as the one-sided-error variant of $1\mathrm{PPDA}_\varepsilon$. In addition, we often use more familiar notation of PCFL, BPCFL, and RCFL respectively for $1\mathrm{PPDA}_{ub}$, $\bigcup_{0\le\varepsilon<1/2} 1\mathrm{PPDA}_\varepsilon$, and $\bigcup_{0\le\varepsilon<1} 1\mathrm{PPDA}_\varepsilon$, while CFL denotes the family of context-free languages. Since $\{a^n b^n c^n \mid n \ge 0\}$ is in BPCFL [5], BPCFL $\not\subseteq$ CFL follows, further leading to PCFL $\neq$ CFL.

For limited automata, $k\text{-}\mathrm{LPA}_\varepsilon$ with an index $k \ge 1$ refers to the family of all languages recognized by k-lpa's with error probability at most ε. Using bounded-error k-lpa's, we denote by k-LBPA the union $\bigcup_{\varepsilon\in[0,1/2)} k\text{-}\mathrm{LPA}_\varepsilon$. In the unbounded-error case, we write k-LPA (or $k\text{-}\mathrm{LPA}_{ub}$ for clarity). Similarly, $k\text{-}\mathrm{LRA}_\varepsilon$ is characterized by one-sided ε-error k-lpa's. Let k-LRA $= \bigcup_{\varepsilon\in[0,1)} k\text{-LRA}$.

Using k-lda's and k-lna's, we define k-LDA and k-LNA, respectively. Pighizzini and Pisoni [11] demonstrated that 2-LDA coincides with DCFL, which is the deterministic variant of CFL. Hibbard [3] proved that k-LNA = CFL for any $k \ge 2$. It is also possible to show that PCFL $\subseteq$ 2-LPA and BPCFL $\subseteq$ 2-LBPA; however, the opposite inclusions are not known to hold. Therefore, our purpose of exact characterizations of PCFL and BPCFL requires a specific property of k-lpa's, called *blank skipping*, for which a k-lpa writes only a unique blank symbol, say, B during the kth visit and it makes only the same deterministic moves while reading B in such a way that it neither changes its inner state nor changes the head direction (either to the right or to the left); in other words, it behaves exactly in the same way while reading consecutive blank symbols. This property plays an essential role in simulating various pushdown automata by limited automata. To emphasize the use of the *blank skipping property*, we append the prefix "bs-", as in $\text{bs-2-LPA}_\varepsilon$. We then obtain the following characterizations.

Theorem 1. *Let $\varepsilon \in [0, 1/2)$ be any error bound.*

1. $2\mathrm{PFA}_\varepsilon = 1\text{-}\mathrm{LPA}_\varepsilon$ *and* $2\mathrm{PFA}_{ub} = 1\text{-}\mathrm{LPA}_{ub}$.
2. $1\mathrm{PPDA}_\varepsilon = \text{bs-2-}\mathrm{LPA}_\varepsilon$, $1\mathrm{RPDA}_\varepsilon = \text{bs-2-}\mathrm{LRA}_\varepsilon$, *and* $\mathrm{PCFL} = \text{bs-2-}\mathrm{LPA}_{ub}$.

Theorem 1(2), in particular, follows from the fact shown in Sect. 3.2 that 1ppda's can be converted into their "ideal shapes."

In the case of k-lda's, as shown in Proposition 2, we can transform limited automata into their blank skipping form and this is, in fact, a main reason that 2-LDA equals DCFL (due to Theorem 1(2) with setting $\varepsilon = 0$).

Proposition 2. *For each index $k \ge 2$, k-LDA = bs-k-LDA.*

For other limited automata, it is not yet clear that, for example, k-LPA = bs-k-LPA.

The second goal of this paper is to argue on various separations of the aforementioned language families. Earlier, Hibbard [3] devised an example language that can separate $(k+1)$-LDA from k-LDA for each index $k \geq 2$. In the case of $k = 2$, a much simpler example language was given in [11]: $L = \{a^n b^n c, a^n b^{2n} d \mid n \geq 0\}$, which is in 3-LDA but not in 2-LDA.

Proposition 3. *For any* $k \geq 2$, k-LDA $\subsetneqq$ k-LRA $\subsetneqq$ k-LBPA $\subseteq$ k-LPA.

Unfortunately, it is unknown whether k-LBPA $\neq$ $(k+1)$-LBPA for each index $k \geq 2$. Proposition 3 will be shown by exploring basic closure properties of target language families. In Sect. 4.4, we will explore these properties in depth.

The language family 2-LRA turns out to be relatively large since it contains languages not recognized by any k-lda for every fixed index $k \geq 2$.

Theorem 4. *For any index* $k \geq 2$, 2-LRA $\not\subseteq$ k-LDA.

Let ω-LDA stand for $\bigcup_{k\geq 1} k$-LDA. Notice that Theorem 4 is not strong enough to yield the separation of 2-LRA $\not\subseteq$ ω-LDA. We also do not know whether or not 3-LDA $\not\subseteq$ 2-LRA and 3-LRA $\not\subseteq$ 2-LBPA.

We seek a refinement of CFL using *unambiguous computation* (i.e., nondeterministic computation with at most one accepting path). Let us define UCFL, from CFL, by restricting 1npda's to have unambiguous computation (see [15]).

Theorem 5. ω-LDA $\subseteq$ UCFL $\subsetneqq$ CFL.

To show Theorem 5, we need to (1) introduce a new model of *k-limited unambiguous automata* (or k-lua's, for short) and its corresponding language family k-LUA, (2) show that k-LUA = bs-k-LUA by a similar argument used for k-LDA, and (3) prove that k-LUA = $(k+1)$-LUA for each index $k \geq 1$ by employing a similar argument for k-LNA. Item (3) then yields a conclusion that ω-LUA $(= \bigcup_{k\geq 1} k$-LUA) equals 2-LUA. Since k-LDA $\subseteq$ k-LUA, we immediately obtain ω-LDA $\subseteq$ 2-LUA = UCFL. This obviously implies Theorem 5. Due to page limit, we omit the details of the above proof.

Wang [13] showed that DCFL contains all languages recognized with bounded-error probability by 2pfa's having rational transition probabilities. Let k-LBPA$^{(rat)}$ denote the subclass of k-LBPA defined by k-lpa's using only rational transition probabilities. Let k-LRA$_{<1/2} = \bigcup_{\varepsilon\in[0,1/2)} k$-LRA$_\varepsilon$. Theorem 1(1) thus implies the following.

Corollary 6. 1-LBPA$^{(rat)}$ $\subseteq$ DCFL $\subseteq$ 2-LRA$_{<1/2}$.

2 Limited Automata

Let us formally introduce various computational models of limited automata, in which we can rewrite the content of each tape cell only during the first k scans or visits of the cell.

Let $\mathbb{N}$ be the set of all non-negative integers and set $\mathbb{N}^+ = \mathbb{N} - \{0\}$. We denote by $[m,n]_{\mathbb{Z}}$ the set $\{m, m+1, m+2, \ldots, n\}$ for any two integers m and n with $m \leq n$. In addition, we abbreviate as $[m]$ the integer interval $[1,m]_{\mathbb{Z}}$ for any integer $m \geq 1$.

2.1 Definitions of k-lpa's with the k-Limitedness Requirement

A *k-limited probabilistic automaton* (or a *k-lpa*, for short) M is formally defined as a tuple $(Q, \Sigma, \{\mathfrak{c}, \$\}, \{\Gamma_i\}_{i\in[k]}, \delta, q_0, Q_{acc}, Q_{rej})$, which accesses only tape area in between two endmarkers (those endmarkers can be accessible but not changeable), where Q is a finite set of (inner) states, Q_{acc} $(\subseteq Q)$ is a set of accepting states, Q_{rej} $(\subseteq Q)$ is a set of rejecting states, Σ is an input alphabet, $\{\Gamma_i\}_{i\in[k]}$ is a collection of mutually disjoint finite sets of tape symbols, q_0 is an initial state in Q, and δ is a probabilistic transition function from $(Q - Q_{halt}) \times \Gamma \times Q \times \Gamma \times D$ to the real unit interval $[0,1]$ with $D = \{-1, +1\}$, $Q_{halt} = Q_{acc} \cup Q_{rej}$, and $\Gamma = \bigcup_{i=0}^{k} \Gamma_i$ for $\Gamma_0 = \Sigma$ and $\mathfrak{c}, \$ \in \Gamma_k$. We implicitly assume that $Q_{acc} \cap Q_{rej} = \emptyset$. The k-lpa has a rewritable tape, on which an input string is initially placed, surrounded by two endmarkers $\mathfrak{c}$ (left endmarker) and \$ (right endmarker). In our formulation of k-lpa, the tape head always moves either to the right or to the left without stopping still. Along each computation path, M probabilistically chooses one of all possible transitions given by δ.

Purely for clarity reason, we express $\delta(q, \sigma, p, \tau, d)$ as $\delta(q, \sigma \mid p, \tau, d)$. Each value $\delta(q, \sigma \mid p, \tau, d)$ indicates the probability that, when M scans σ on the tape in inner state q, M changes its inner state to p, overwrites τ onto σ, and moves its tape head in direction d. We set $\delta[q, \sigma] = \sum_{(p,\tau,d)\in Q\times\Gamma\times D} \delta(q, \sigma \mid p, \tau, d)$. The function δ must satisfy $\delta[q, \sigma] = 1$ for every pair $(q, \sigma) \in Q \times \Gamma$.

The k-lpa M must satisfy the following *k-limitedness requirement*: during the first k scans of each tape cell, at the ith scan with $0 \leq i < k$, if M reads the content of the cell containing a symbol in Γ_i, then M rewrites it to another symbol in Γ_{i+1}. After the the kth scan, the cell becomes unchangeable (or *frozen*); that is, M still reads a symbol in the cell but M no longer alters the symbol. For the above rule, there is one exception: whenever the tape head *makes a turn* (either from the left to the right or from the right to the left) at any tape cell, we count this move as "double scans" or "double visits." To make the endmarkers special, we assume that no symbol in $\bigcup_{i=}^{k-1} \Gamma_i$ can be replaced by any endmarker. This k-limitedness requirement is formally stated as follows: for any transition $\delta(q, \sigma \mid p, \tau, d) \neq 0$ with $p, q \in Q$, $\sigma \in \Gamma_i$, $\tau \in \Gamma_j$, and $d \in \{+1, -1\}$, (1) if $i = k$, then $\sigma = \tau$ and $j = i$, (2) if $i < k$ and i is even, then $j = i + 2^{(1-d)/2}$, and (3) if $i < k$ and i is odd, then $j = i + 2^{(1+d)/2}$.

The probability of each computation path is determined by the multiplication of all chosen transition probabilities along the path. The *acceptance probability* of M on input x is the sum of all probabilities of accepting computation paths of M starting with the input x. We express by $p_{M,acc}(x)$ the total acceptance probability of M on x. Similarly, we define $p_{M,rej}(x)$ to be the *rejection probability* of M on x. Given a k-lpa M, we say that M *accepts* x if $p_{M,acc}(x) > 1/2$ and

rejects x if $p_{M,rej}(x) \geq 1/2$. The notation $L(M)$ stands for the set of all strings x accepted by M. Given a language L, we say that M *recognizes* L exactly when $L = L(M)$. We further say that M makes *bounded error* if there exists a constant $\varepsilon \in [0, 1/2)$ (called an *error bound*) such that, for every input x, either $p_{M,acc}(x) \geq 1 - \varepsilon$ or $p_{M,rej}(x) \geq 1 - \varepsilon$. With or without this condition, M is said to make *unbounded error*. For a language L, the *error probability of* M *on* x *for* L is the probability that M's outcome is different from L.

Generally, a k-lpa may produce an extremely long computation path or even an infinite computation path. Following an early discussion in Sect. 1.1 on the expected runtime of probabilistic machines, it is desirable to restrict our attention to k-lpa's whose *computation paths have a polynomial length on average*; that is, there is a polynomial p for which the expected length of all terminating computation paths on input x is bounded from above by $p(|x|)$. In what follows, we implicitly assume that all k-lpa's should satisfy this expected polynomial termination requirement. Given an input x, we say that M *accepts (resp., rejects)* x *with probability* p if the total probability of accepting (resp., rejecting) computation paths is exactly p.

Let us recall the language families introduced in Sect. 1.2, associated with limited automata. Among these language families, for each index $k \geq 2$, it follows from the above definitions and by [3] that k-LDA $\subseteq$ k-LRA$_\varepsilon$ $\subseteq$ 2-LNA $=$ CFL and k-LBPA$_{\varepsilon'} \subseteq k$-LPA$_{\varepsilon'}$ for any constants $\varepsilon \in [0, 1)$ and $\varepsilon' \in [0, 1/2)$. Moreover, by amplifying the success probability of k-lra's, we easily obtain the inclusion: k-LRA $\subseteq$ k-LBPA for every index $k \geq 1$.

3 One-Way Pushdown Automata

We will formally describe various one-way pushdown automata.

3.1 One-Way Probabilistic Pushdown Automata

One-way deterministic and nondeterministic pushdown automata (abbreviated as 1dpda's and 1npda's, respectively) can be viewed as special cases of the following *one-way probabilistic pushdown automata* (or *1ppda's*, for short).

Formally, a 1ppda M is a tuple $(Q, \Sigma, \{¢, \$\}, \Gamma, \Theta_\Gamma, \delta, q_0, Z_0, Q_{acc}, Q_{rej})$, in which Q is a finite set of (inner) states, Σ is an input alphabet, Γ is a stack alphabet, Θ_Γ is a finite subset of Γ^* with $\lambda \in \Theta_\Gamma$, δ is a *probabilistic transition function* (with $\check{\Sigma} = \Sigma \cup \{\lambda, ¢, \$\}$) from $(Q - Q_{halt}) \times \check{\Sigma} \times \Gamma \times Q \times \Theta_\Gamma$ to $[0, 1]$, q_0 ($\in Q$) is an initial state, Z_0 ($\in \Gamma$) is a bottom marker, Q_{acc} ($\subseteq Q$) is a set of accepting states, and Q_{rej} ($\subseteq Q$) is a set of rejecting states, where λ is the empty string and $Q_{halt} = Q_{acc} \cup Q_{rej}$.

For clarity reason, we express $\delta(q, \sigma, a, p, u)$ as $\delta(q, \sigma, a \mid p, u)$. Let $\delta[q, \sigma, a] = \sum_{(p,u) \in Q \times \Theta_\Gamma} \delta(q, \sigma, a \mid p, u)$ with $\sigma \in \check{\Sigma}$. When $\sigma = \lambda$, we call its transition a λ-*move* (or a λ-*transition*) and the tape head must stay still. At any point, M can probabilistically select either a λ-move or a non-λ-move. This is formally stated as $\delta[q, \sigma, a] + \delta[q, \lambda, a] = 1$ for any given tuple $(q, \sigma, a) \in (Q - Q_{halt}) \times (\Sigma \cup \{¢, \$\}) \times$

Γ. In a way similar to k-lpa's, we can define the notions of unbounded-error, bounded-error, acceptance/rejection probability, etc. We require every 1ppda to run *in expected polynomial time*. Two 1ppda's M_1 and M_2 are *(recognition) equivalent* if $L(M_1) = L(M_2)$. Let us recall the language families described in Sect. 1.2. It is well-known that DCFL $\subseteq$ BPCFL $\subseteq$ PCFL.

For two language families $\mathcal{F}$ and $\mathcal{G}$, the notation $\mathcal{F} \vee \mathcal{G}$ (resp., $\mathcal{F} \wedge \mathcal{G}$) denotes the *2-disjunctive closure* $\{A \cup B \mid A \in \mathcal{F}, B \in \mathcal{G}\}$ (resp., the *2-conjunctive closure* $\{A \cap B \mid A \in \mathcal{F}, B \in \mathcal{G}\}$). For any index $d \in \mathbb{N}^+$, define $\mathcal{F}(1) = \mathcal{F}$ and $\mathcal{F}(d+1) = \mathcal{F} \wedge \mathcal{F}(d)$. Notice that $\mathrm{CFL}(k) \neq \mathrm{CFL}(k+1)$ for any $k \in \mathbb{N}^+$ [8].

3.2 An Ideal Shape of 1ppda's

We want to show how to convert any 1ppda to a "pop-controlled form" (called an *ideal shape*), in which the pop operations always take place by first reading an input symbol σ and then making a series (one or more) of the pop operations without reading any further input symbol. In other words, a 1ppda *in an ideal shape* is restricted to take only the following transitions. Let $\Gamma^{(-)} = \Gamma - \{Z_0\}$. (1) Scanning $\sigma \in \Sigma$, preserve the topmost stack symbol (called a *stationary operation*). (2) Scanning $\sigma \in \Sigma$, push a new symbol u ($\in \Gamma^{(-)}$) without changing any other symbol in the stack. (3) Scanning $\sigma \in \Sigma$, pop the topmost stack symbol. (4) Without scanning an input symbol (i.e., λ-move), pop the topmost stack symbol. (5) The stack operations (4) comes only after either (3) or (4).

Lemma 7 states that any 1ppda can be converted into its "equivalent" 1ppda in an ideal shape. We say that two 1ppda's are *error-equivalent* if they are recognition equivalent and their error probabilities coincide on all inputs. The *push size* of a 1ppda is the maximum length of any string pushed into a stack by any single move.

Lemma 7 (Ideal Shape Lemma for 1ppda's). *Let $n \in \mathbb{N}^+$. Any n-state 1ppda M with stack alphabet size m and push size e can be converted into another error-equivalent 1ppda N in an ideal shape with $O(en^2m^2(2m)^{2enm})$ states and stack alphabet size $O(enm(2m)^{2enm})$. This is true for the model with no end-marker.*

The proof of this lemma is lengthy, consisting of a series of transformations of automata, and is proven by utilizing, in part, ideas of Hopcroft and Ullman [4, Chap. 10] and of Pighizzini and Pisoni [11, Sect. 5].

The ideal shape lemma is useful for simplifying certain proofs. In what follows, we give one such example. Given a language A, the notation A^R denotes the *reverse language* $\{x^R \mid x \in A\}$. For a family $\mathcal{F}$ of languages, $\mathcal{F}^R$ expresses the collection of A^R for any language A in $\mathcal{F}$.

Lemma 8. PCFL *is closed under reversal; that is,* $\mathrm{PCFL}^R = \mathrm{PCFL}$.

4 Behaviors of Limited Automata

In the subsequent subsections, we intend to verify our main results stated in Sect. 1.2 by making structural analyses on the behaviors of k-lpa's.

4.1 Blank Skipping Property, Theorem 1, and Proposition 2

We will show the proofs of Theorem 1 and Proposition 2. For the former proof, we want to restrict the behaviors of k-lpa's so that we can control their computation. Firstly, we give the formal description of the notion of blank skipping property. A k-lpa is *blank skipping* if (1) $\Gamma_k = \{¢, \$, B\}$, where B is a unique blank symbol, and (2) there are two disjoint subsets Q_{+1}, Q_{-1} of Q for which $\delta(q, B \mid q, B, d) = 1$ for any direction $d \in \{\pm 1\}$ and any state $q \in Q_d$. In other words, when a k-lpa passes a cell for the kth time, it must make the cell *blank* (i.e., the cell has B) and the cell becomes frozen afterward.

Let us begin with the proof of Theorem 1.

Proof Sketch of Theorem 1. (1) It is rather easy to simulate a 2pfa by a 1-lpa that behaves like the 2pfa but changes each input symbol σ to its corresponding symbol σ'. On the contrary, we want to simulate a 1-lpa M by the following 2pfa N. A key idea is that it is possible to maintain and update a list of state pairs, each (p, q) of which indicates that, if M's tape head enters the tape area left of the currently scanning cell from the right in state p, then M eventually leaves the area to the right in state q with positive probability.

(2) This directly comes from Lemmas 9 and 10. □

In what follows, we describe Lemmas 9 and 10 and present their proofs.

Lemma 9. *Let $n \geq 2$ and $l \geq 1$. Every n-state blank-skipping 2-lpa working on an l-letter alphabet can be converted into a recognition-equivalent 1ppda with the same error probability and of states at most $2n$.*

Proof Sketch. Given a blank-skipping 2-lpa M, we simulate it as follows. On input x, when M reads a new input symbol σ by changing it to τ, we read σ and push τ into a stack. In contrast, when M moves its tape head leftwards by skipping B to the first non-blank symbol τ and changes it to B, we simply pop a topmost stack symbol. This simulation can be done by a certain 1ppda. □

The ideal shape form of 1ppda's is a key to the next lemma.

Lemma 10. *Let $n, l \in \mathbb{N}^+$. Let L be a language over an alphabet Σ of size l recognized by an n-state 1ppda M in an ideal shape with error probability at most ε. There is a blank-skipping 2-lpa N that has $O(nl)$ states and recognizes L with the same error probability.*

Proof Sketch. Let M be a 1ppda and assume that M is in an ideal shape. We simulate M by an appropriate 2-lpa in the following way. Let x be any input string. Assume that M reads a new input symbol σ. If M pushes τ into a stack, then we read σ and change it into τ. If M pops a topmost stack symbol, then we move a tape head leftwards to read the first non-blank symbol τ and then replace it with B. On the contrary, assume that M makes a λ-move. Since M's move must be a pop operation, we move the tape head leftwards and replace the first non-blank symbol by B. □

It is possible to convert any k-lda into its equivalent blank-skipping k-lda. The following is a key lemma, from which Proposition 2 follows immediately. Our proof partly takes an idea from [11].

Lemma 11. *Let k be any integer with $k \geq 2$. Given any k-lda M, there exists another k-lda N such that (1) N is blank-skipping and (2) N agrees with M on all inputs.*

4.2 Properties of ω-LPFA

As done in [14–16], we equip each 1nfa with a *write-only output tape*.[2] Let 1NFAMV denote the class of all multi-valued partial functions from Σ_1^* to Σ_2^* whose output values are produced on write-only tapes along only accepting computation paths of 1nfa's, where Σ_1 and Σ_2 are arbitrary alphabets. We further write $1\mathrm{NFAMV}_t$ for the collection of all *total* functions in 1NFAMV. Let $k \geq 2$. For any $f : \Sigma_1^* \to \Sigma_2^*$ in $1\mathrm{NFAMV}_t$ witnessed by a 1nfa, say, M_f with an output tape and for any k-lpa M over Σ_2, let $L_{f,M} = \{x \in \Sigma_1^* \mid \sum_{y \in \Sigma_2^*} |AP_f(x|y)|\mathrm{Prob}_M[M(y) = 1]/|AP_f(x)| > 1/2\}$, where $AP_f(x|y)$ is the set of all accepting computation paths of M_f producing y on input x and $AP_f(x) = \bigcup_{y \in \Sigma_2^*} AP_f(x|y)$. By abusing the notation, we write $k\text{-LPA} \circ 1\mathrm{NFAMV}_t$ to denote the set of all such $L_{f,M}$'s.

We argue that k-LPA is "invariant" with an application of $1\mathrm{NFAMV}_t$-functions in the following sense.

Lemma 12. *For any index $k \geq 2$, $k\text{-LPA} \circ 1\mathrm{NFAMV}_t = k\text{-LPA}$.*

Proof Sketch. Let $k \geq 2$. Since it is obvious that $k\text{-LPA} \subseteq k\text{-LPA} \circ 1\mathrm{NFAMV}_t$, we want to show the opposite inclusion. Take a function $f \in 1\mathrm{NFAMV}_t$ and a k-lpa M, and consider $L_{f,M}$. There is a 1nfa M_f computing f. Consider the following machine N. On input x, run M_f and, whenever M_f produces one output symbol σ, run M to process σ. Along each computation path of M_f, if M_f enters an accepting state, then N does the same, otherwise, N enters both accepting and rejecting states with equiprobability. Clearly, N is also a k-lpa and it recognizes L with unbounded-error probability. □

Consider any k-lpa M used in the definition of $k\text{-LPA} \circ 1\mathrm{NFAMV}_t$. If we feed such an M with the reverses x^R of inputs x, then we obtain $k\text{-LPA}^R \circ 1\mathrm{NFAMV}_t$. We show the following relationship between $(k+1)$-LPA and $k\text{-LPA}^R$.

Lemma 13. *For any $k \geq 2$, $k\text{-LPA}^R \circ 1\mathrm{NFAMV}_t \subseteq (k+1)\text{-LPA}$.*

Proof Sketch. Fix $k \geq 2$.

We show the inclusion of $k\text{-LPA}^R \circ 1\mathrm{NFAMV}_t \subseteq (k+1)\text{-LPA}$. Let M be a k-lpa and let f be a function in $1\mathrm{NFAMV}_t$ witnessed by a certain 1nfa, say, M_f. Define $L_{f,M^R} = \{x \in \Sigma^* \mid \sum_y |AP_f(x|y)|\mathrm{Prob}_M[M(y^R) = 1]/|AP_f(x)| > 1/2\}$.

[2] An output tape is *write only* if its cells are initially blank and its tape head moves to the right whenever it writes a non-blank symbol.

Our goal is to verify that $L_{f,M^R} \in (k+1)$-LPA. Consider the following machine N. On input x, run M_f on x, change x into y in $f(x)$ along all computation paths in $AC_f(x|y)$, and run M on $\mathbb{k}y\$$ starting at \$ and ending at $\mathbb{k}$. Since M is k-limited, N must be $(k+1)$-limited. We thus conclude that $L_{f,M^R} \in (k+1)$-LPA. □

4.3 Power of Probabilistic Computation and Theorem 4

We will give the proof of Theorem 4. The proof requires, for each index $k \geq 2$, a certain language, which is in $(k+1)$-LDA but outside of k-LDA. The example languages shown below are slight modifications of the ones given by Hibbard [3].

(1) When $k = 3$, the language L_3 is composed of all strings of the forms $a^{n_2}b^{n_2}c^{p_2}\#a$ and $a^{n_2}b^{m_2}c^{m_2}\#b$ for all integers $n_2, m_2, p_2 \geq 0$.

(2) Let $k \geq 4$ be any index and assume that L_{k-1} is already defined. For each index $i \in [2, k-1]_{\mathbb{Z}}$, we succinctly write w_i in place of $a^{n_i}b^{m_i}c^{p_i}$ for certain numbers $n_i, m_i, p_i \in \mathbb{N}$. The desired language L_k is composed of all strings w of the form $w_2\#w_4\#\cdots\#w_{k-1}\#\cdots\#w_5\#w_3\#x$ with $x \in \{a,b\}$ satisfying Conditions (i)–(iv) given below. For each index $i \in [3, k-2]_{\mathbb{Z}}$, we define $\tilde{w}_i^{(k)} = w_{i-1}$ if i is even, and w_{i+1} otherwise. Moreover, let $\tilde{w}_{k-1}^{(k)} = w_{k-1}$ if k is even, and w_{k-2} otherwise. Finally, let $\tilde{w}^{(-)}$ express the string $\tilde{w}_4^{(k)}\#\tilde{w}_6^{(k)}\#\cdots\#\tilde{w}_{k-1}^{(k)}\#\cdots\#\tilde{w}_5^{(k)}\#\tilde{w}_3^{(k)}$. (i) If $x = a$ and $n_2 = m_2$, then $\tilde{w}^{(-)}\#a \in L_{k-1}$. (ii) If $x = a$ and $n_2 < m_2$, then $\tilde{w}^{(-)}\#b \in L_{k-1}$. (iii) If $x = b$ and $m_2 = p_2$, then $\tilde{w}^{(-)}\#a \in L_{k-1}$. (iv) If $x = b$ and $m_2 < p_2$, then $\tilde{w}^{(-)}\#b \in L_{k-1}$.

An argument similar to [3, Sect. 4] verifies that, for each index $k \geq 2$, the language L_{k+1} is included in $(k+1)$-LDA but excluded from k-LDA.

Fix $k \geq 2$. For each symbol $x \in \{a,b\}$, let $L_x = \{w\#x \in L_{k+1} \mid w = w_2\#w_4\#\cdots\#w_k\#\cdots\#w_5\#w_3\}$. Note that $L_{k+1} = L_a \cup L_b$. Since $L_a, L_b \in k$-LDA, it follows that $L_{k+1} \in k\text{-LDA} \vee k\text{-LDA}$. Therefore, we obtain the following corollary.

Corollary 14. *For every $k \geq 2$, k-LDA $\vee$ k-LDA $\not\subseteq$ k-LDA.*

Lemma 15. *For any $k \geq 3$, $L_k \in 2\text{-LRA}_{(1-2^{-2k+5})}$.*

Proof. We first show that $L_3 \in 2\text{-LRA}_{1/2}$. Let $L'_a = \{a^n b^n c^p \# a \mid n, p \geq 0\}$ and $L'_b = \{a^n b^m c^m \# b \mid n, m \geq 0\}$. Clearly, $L_3 = L'_a \cup L'_b$ holds. Since $L'_a, L'_b \in$ DCFL, for each symbol $x \in \{a,b\}$, we take a 2-lda M_x that recognizes L'_x. Consider the following 2-lra N. Let w be any input. Initially, choose an index $x \in \{a,b\}$ with equiprobability and then run M_x. If $w \in L_3$, then N accepts w with probability $1/2$; otherwise, N rejects w with probability 1.

By induction hypothesis, we assume that $L_k \in 2\text{-LRA}_{(1-2^{-2k+5})}$. Let us consider L_{k+1}. Using the aforementioned notation, define $L_{a=} = \{w\#a \mid n_2 = m_2, \tilde{w}^{(-)}\#a \in L_k\}$ and $L_{a<} = \{w\#a \mid n_2 < m_2, \tilde{w}^{(-)}\#a \in L_k\}$, where w is of the form $w_2\#w_4\#\cdots\#w_k\#\cdots\#w_5\#w_3\#x$ for a certain symbol $x \in \{a,b\}$. Similarly, we define $L_{b=}$ and $L_{b<}$. Note that $L_{k+1} = L_{a=} \cup L_{a<} \cup L_{b=} \cup L_{b<}$. It is not difficult to show that $L_{a=}, L_{a<}, L_{b=}, L_{b<}$ all belong to $2\text{-LRA}_{(1-2^{-2k+5})}$ since

$L_k \in 2\text{-LRA}_{(1-2^{-2k+5})}$. Consider the following machine N. On the input of the form $w\#x$, choose one of the pairs $\{a =, a <, b =, b <\}$ with equal probability. Suppose that we have chosen $a =$. As an example, let $M_{a=}$ be a 2-lra recognizing the language $L_{a=}$. In this case, run $M_{a=}$ on $w\#x$. When $w\#a \in L_{k+1}$, N accepts the input with probability $\frac{1}{4} \times 2^{-2k+5}$, which equals $2^{-2(k+1)+5}$. The other cases are similarly treated. □

Lemma 15 implies that $L_{k+1} \in 2\text{-LRA}$. Since $L_{k+1} \notin k\text{-LDA}$, we instantly conclude that $2\text{-LRA} \not\subseteq k\text{-LDA}$. This completes the proof of Theorem 4.

4.4 Closure Properties of Probabilistic Classes and Proposition 3

We will explore basic closure properties of k-LRA, k-LBPA, and k-LDA. By utilizing some of those properties, we will prove Proposition 3 in the end.

Lemma 16. *For any $k \geq 2$, k-LRA is closed under finite union but not under finite intersection.*

Lemma 17. *For any $k \geq 2$, k-LBPA is closed under complementation but k-LRA is not.*

Proof. It is not difficult to show that $k\text{-LBPA} = \text{co-}k\text{-LBPA}$ for all indices $k \geq 2$. By Lemma 16, k-LRA is closed under finite union. If $k\text{-LRA} = \text{co-}k\text{-LRA}$, then k-LRA must be closed under finite intersection. This contradicts the second part of Lemma 16. □

Recall that $2\text{-LDA} = \text{DCFL}$ [11]. Although $k\text{-LDA} \neq \text{DCFL}$ for all $k \geq 3$, k-LDA still satisfies many non-closure properties as DCFL does.

Lemma 18. *For any $k \geq 2$, k-LDA is not closed under reversal, concatenation, λ-free homomorphism, or Kleene star.*

Next, we look at the closure properties of k-LBPA.

Lemma 19. *For each operator $\diamond \in \{\wedge, \vee\}$, $k\text{-LDA} \diamond k\text{-LDA} \subseteq k\text{-LBPA}$; thus, $k\text{-LDA}(2) \subseteq k\text{-LBPA}$.*

Proof. It suffices to consider the case of $\diamond = \vee$ because k-LBPA is closed under complementation. Let M_1, M_2 be two k-lpa's working over the same alphabet Σ. We design a new k-lpa N to work as follows. On input x, choose an index $i \in \{1, 2\}$ uniformly at random, run M_i on x. If M_i enters an accepting state, accept x with probability 1; otherwise, accept x with probability $1/3$ and reject with probability $2/3$. If $x \in L(M_1) \cup L(M_2)$, then the acceptance probability of N is at least $2/3$. In contrast, if $x \notin L(M_1) \cup L(M_2)$, then the rejection probability is at least $2/3$. Therefore, $L(M_1) \cup L(M_2)$ is in k-LBPA. □

It is, however, unknown that $k\text{-LDA}(d) \subseteq k\text{-LBPA}$ for every index $d \geq 3$.

Proof of Proposition 3. All inclusions obviously hold. We want to show the remaining two separations. Note that $k\text{-LDA} = \text{co-}k\text{-LDA}$ for any $k \geq 1$. By Lemma 16, it follows that $k\text{-LDA} \neq k\text{-LRA}$. Similarly, from $k\text{-LBPA} = \text{co-}k\text{-LBPA}$, we obtain $k\text{-LRA} \neq k\text{-LBPA}$. □

References

1. Dwork, C., Stockmeyer, L.: Finite verifiers I: the power of interaction. J. ACM **39**, 800–828 (1992)
2. Freivalds, R.: Probabilistic two-way machines. In: Gruska, J., Chytil, M. (eds.) MFCS 1981. LNCS, vol. 118, pp. 33–45. Springer, Heidelberg (1981). https://doi.org/10.1007/3-540-10856-4_72
3. Hibbard, T.N.: A generalization of context-free determinism. Inf. Control **11**, 196–238 (1967)
4. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston (1979)
5. Hromkovič, J., Schnitger, G.: On probabilistic pushdown automata. Inf. Comput. **208**, 982–995 (2010)
6. Kaŋeps, J., Freivalds, R.: Minimal nontrivial space complexity of probabilistic one-way turing machines. In: Rovan, B. (ed.) MFCS 1990. LNCS, vol. 452, pp. 355–361. Springer, Heidelberg (1990). https://doi.org/10.1007/BFb0029629
7. Kaŋeps, J., Geidmanis, D., Freivalds, R.: Tally languages accepted by Monte Carlo pushdown automata. In: Rolim, J. (ed.) RANDOM 1997. LNCS, vol. 1269, pp. 187–195. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63248-4_16
8. Liu, L.Y., Weiner, P.: An infinite hierarchy of intersections of context-free languages. Math. Syst. Theory **7**, 185–192 (1973)
9. Macarie, I.I., Ogihara, M.: Properties of probabilistic pushdown automata. Theor. Comput. Sci. **207**, 117–130 (1998)
10. Pighizzini, G., Pisoni, A.: Limited automata and regular languages. Int. J. Found. Comput. Sci. **25**, 897–916 (2014)
11. Pighizzini, G., Pisoni, A.: Limited automata and context-free languages. Fund. Inform. **136**, 157–176 (2015)
12. Wagner, K., Wechsung, G.: Computational Complexity. D. Reidel Publishing, Dordrecht (1986)
13. Wang, J.: A note on two-way probabilistic automata. Inf. Process. Lett. **43**, 321–326 (1992)
14. Yamakami, T.: Oracle pushdown automata, nondeterministic reducibilities, and the hierarchy over the family of context-free languages. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 514–525. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04298-5_45
15. Yamakami, T.: Structural complexity of multi-valued partial functions computed by nondeterministic pushdown automata. In: Proceedings of ICTCS 2014, CEUR Workshop Proceedings, vol. 1231, pp. 225–236 (2014)
16. Yamakami, T.: Not all multi-valued partial CFL functions are refined by single-valued functions (extended abstract). In: Diaz, J., Lanese, I., Sangiorgi, D. (eds.) TCS 2014. LNCS, vol. 8705, pp. 136–150. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44602-7_12
17. Yamakami, T.: One-way bounded-error probabilistic pushdown automata and kolmogorov complexity (preliminary report). In: Charlier, É., Leroy, J., Rigo, M. (eds.) DLT 2017. LNCS, vol. 10396, pp. 353–364. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62809-7_27