Unambiguous and Co-Nondeterministic Computations of Finite Automata and Pushdown Automata Families and the Effects of Multiple Counters

Tomoyuki Yamakami[1]

Abstract

Nonuniform families of polynomial-size finite automata and pushdown automata respectively have strong connections to nonuniform-NL and nonuniform-LOGCFL. We examine the behaviors of unambiguous and co-nondeterministic computations produced by such families of automata operating multiple counters. As its consequences, we obtain various collapses of the complexity classes of families of promise problems solvable by finite and pushdown automata families when all valid instances are limited to either polynomially long strings or unary strings. A key technical ingredient of our proofs is an inductive counting of reachable vertices of each computation graph of finite and pushdown automata that operate multiple counters simultaneously.

1 Background and Challenging Questions

This section will provide background knowledge on the topics of this work, raise important open questions, and give their solutions.

1.1 Two Important Open Questions in Nonuniform Polynomial State Complexity Theory

Nondeterministic computation has played an important role in computational complexity theory as well as automata theory. Associated with such computation, there are two central and crucial questions to resolve. (i) Can any co-nondeterministic computation be simulated on an appropriate nondeterministic machine? (ii) Can any nondeterministic machine be made unambiguous? In the polynomial-time setting, these questions correspond to the famous NP =?co-NP and NP =?UP questions. In this work, we attempt to resolve these questions in the setting of nonuniform polynomial state complexity classes.

We quickly review the origin and the latest progress of the study of nonuniform state complexity classes. Apart from a standard uniform model of finite automata, Berman and Lingas [1] and Sakoda and Sipser [14] considered, as a "collective" model of computations, nonuniform families of two-way finite automata indexed by natural numbers and they studied the computational power of these families of finite automata having *polynomial size* (i.e., having polynomially many inner states). Unlike Boolean circuit families, these automata are allowed to take "arbitrarily" long inputs. Of those families of polynomial-size finite automata, Sakoda and Sipser focused on the models of two-way deterministic finite automata (or 2dfa's, for short) and two-way nondeterministic finite automata (or 2nfa's). They introduced the complexity classes, dubbed as 2D and 2N, which consist of all families of "promise" problems[2] solvable respectively by nonuniform families of polynomial-size 2dfa's and 2nfa's. As their natural extensions, nonuniform families of two-way deterministic pushdown automata (or 2dpda's) and two-way nondeterministic pushdown automata (or 2npda's) running in polynomial time have also been studied lately [17, 21]. Similarly to 2D and 2N, these pushdown automata models induce two corresponding complexity classes of promise problem families, denoted respectively by 2DPD and 2NPD. Since an introduction of nonuniform polynomial-size finite automata families, various machine types (such as deterministic, nondeterministic, alternating, probabilistic, and quantum) have been studied in depth [3, 8, 9, 10, 11, 16, 17, 18, 19, 21].

An importance of polynomial-size finite automata families comes from their close connection to decision problems in the nonuniform variants of the log-space complexity classes L and NL, when all promised (or valid) instances given to underlying finite automata are limited to polynomially long strings (where

[1]Present Affiliation: Faculty of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan

[2]A *promise problem* over alphabet Σ is a pair (A, R) satisfying that $A, R \subseteq \Sigma^*$ and $A \cap R = \varnothing$. A *language* L over Σ can be identified with a unique promise problem having the form $(L, \Sigma^* - L)$.

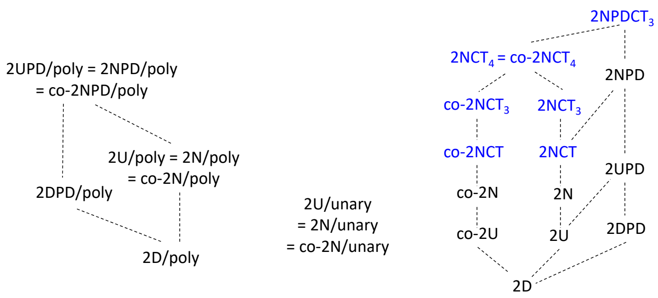

Figure 1: Containments among nonuniform polynomial state complexity classes shown in this work. Remark that the collapse 2U/poly = 2N/poly comes from [19] and 2U/unary = 2N/unary = co-2N/unary are drawn from [4, 5] in Section 4.

this condition is referred to as a *polynomial ceiling*) [10], or when all valid instances are limited to unary strings [11]. In a similar fashion, 2DPD and 2NPD are closely related to the nonuniform versions of LOGDCFL and LOGCFL [17] when all valid instances are only polynomially long, where LOGCFL (resp., LOGDCFL) is the collection of all languages log-space many-one reducible to context-free (resp., deterministic context-free) languages. These strong correspondences to standard computational complexity theory provide one of the good reasons to investigate the fundamental properties of various types of nonuniform automata families in hopes of achieving a better understanding of parallel complexity classes, such as L, NL, LOGDCFL, and LOGCFL, in the nonuniform setting.

For nonuniform finite and pushdown automata families, nevertheless, the aforementioned two central open questions (i)–(ii) correspond to the 2N =?2U, 2N =?co-2N, 2NPD =?2UPD, and 2NPD =?co-2NPD questions. Unfortunately, these four equalities are not known to hold at this moment. It is therefore desirable to continue the study on the behaviors of finite and pushdown automata families in order to deepen our understandings of these machine families and to eventually resolve those four central questions.

1.2 New Challenges and Main Contributions

The primary purpose of this work is to present "partial" solutions to the four important questions raised in Section 1.1 associated with 2N and 2NPD by studying the computational power of nonuniform families of polynomial-size finite and pushdown automata in depth.

In the course of our study, we further look into the key role of "counters", each of which is essentially a stack manipulating only a single symbol, say, "1" except for the bottom marker $\perp$. Since the total number of 1s in a counter can be viewed as a natural number, the counter is able to "count", as its name suggests. For the first time, we supplement multiple counters to the existing models of nonuniform finite and pushdown automata families. We remark that, for short runtime computation, counters are significantly weaker[3] in functionality than full-scale stacks. Even though, the proper use of counters can help us not only trace the tape head location but also count the number of steps.

By appending multiple counters to finite and pushdown automata, we obtain the machine models of *counter automata* and *counter pushdown automata*. Two additional complexity classes, $2\mathrm{NCT}_k$ and $2\mathrm{NPDCT}_k$ are naturally obtained by taking families of polynomial-size nondeterministic counter automata and counter pushdown automata operating k counters.

The use of multiple counters makes it possible to show in Section 3 that 2N and co-2N coincide and that 4 counters are enough (namely, $2\mathrm{NCT}_4 = \text{co-}2\mathrm{NCT}_4$). This result further leads to the equivalence between co-2N and 2N in Section 4.3 when all promise problem families are restricted to having polynomial ceilings. Under the same restriction, we will show that co-2NPD and 2NPD also coincide. Our results are briefly summarized in Fig. 1, in which the suffix "/poly" refers to the polynomial ceiling restriction and the suffix "/unary" refers to the restriction to unary input strings. To obtain some of the equalities in the figure, we will exploit a close connection between *parameterized decision problems* and families of promise problems, which was first observed in [18] and then fully developed in [17, 19, 21].

[3]With exponential overhead, 2-counter automata can simulate a Turing machine [12]

2 Preliminaries: Notions and Notation

We briefly explain fundamental notions and notation used in the rest of this work.

2.1 Numbers, Languages, and Pushdown Automata

The set of all *natural numbers* (including 0) is denoted $\mathbb{N}$ and the positive-integer set $\mathbb{N}-\{0\}$ is expressed as $\mathbb{N}^+$. Given two integers m and n with $m \leq n$, the notation $[m,n]_{\mathbb{Z}}$ denotes the *integer set* $\{m, m+1, m+2, \ldots, n\}$. As a special case, we write $[n]$ for $[1,n]_{\mathbb{Z}}$ when $n \in \mathbb{N}^+$. In this work, all *polynomials* must have nonnegative coefficients and all *logarithms* are taken to the base 2 with the notation $\log 0$ being treated as 0. The *power set* of a set S is denoted $\mathcal{P}(S)$.

Given an alphabet Σ and a number $n \in \mathbb{N}$, the notation Σ^n (resp., $\Sigma^{\leq n}$) denotes the set of all strings over Σ of length exactly n (resp., at most n). The *empty string* is always denoted ε.

As a basic machine model, we use *two-way nondeterministic finite automata* (or 2nfa's, for short) that make neither ε-moves[4] nor stationary moves. This means that input tape heads of 2nfa's always move to adjacent tape cells without stopping at any tape cells. Given a finite automaton M, the *state complexity* $sc(M)$ of M is the total number of inner states used for M.

Another important machine model is *two-way nondeterministic pushdown automata* (or 2npda's) N over alphabet Σ with a stack alphabet Γ including the bottom marker $\perp$. Notice that N is allowed to make *ε-moves* whereas 2nfa's make no ε-moves. The *stack-state complexity* $ssc(N)$ of N denotes the product $|Q| \cdot |\Gamma^{\leq e}|$, which turns out to be a useful complexity measure [17, 21], where Q is a set of inner states and e is the *push size* (i.e., the maximum length of pushed strings into the stack at any single push operation).

A *counter* is a special kind of (pushdown) stack whose alphabet consists only of a single symbol, say, "1" except for $\perp$. In this work, we freely equip multiple counters to finite automata and pushdown automata. These machines are respectively called *counter automata* and *counter pushdown automata*. We conveniently abbreviate a *two-way nondeterministic counter automaton* as a 2ncta and a *two-way nondeterministic counter pushdown automaton* as a 2npdcta.

2.2 Promise Problems and Nonuniform Families

Unlike the notion of languages, *promise problems* over alphabet Σ are formally of the form (A, R) satisfying that $A, R \subseteq \Sigma^*$ and $A \cap R = \varnothing$. We say that a 1nfa (1ncta, 1npda, or 1npdcta) M *solves* (A, R) if (i) for any $x \in A$, M accepts x (i.e., there exists an accepting computation path of M on x) and (ii) for any $x \in R$, M rejects x (i.e., all computation paths of M on x are rejecting). Any string in $A \cup R$ is said to be *promised* or *valid*. Since we do not impose any further condition on all strings outside of $A \cup R$, it suffices to focus only on the promised strings in our later discussion.

For any nondeterministic machine models discussed in Section 2.1, a machine is said to be *unambiguous* if it has at most one accepting computation path on each promised instance. For other instances, there is no restriction on the number of accepting/rejecting computation paths.

Throughout this work, we consider any "family" $\mathcal{L}$ of promise problems $(L_n^{(+)}, L_n^{(-)})$ over a common fixed alphabet Σ indexed by natural numbers $n \in \mathbb{N}$. Such a family $\mathcal{L}$ is said to have a *polynomial ceiling* if there exists a polynomial p such that $L_n^{(+)} \cup L_n^{(-)} \subseteq \Sigma^{\leq p(n)}$ holds for all indices $n \in \mathbb{N}$. Given a complexity class $\mathcal{C}$ of families of promise problems, if we restrict our attention to only promise problem families in $\mathcal{C}$ having a polynomial ceiling, then we obtain the subclass of $\mathcal{C}$, expressed as $\mathcal{C}/\mathrm{poly}$. Moreover, when all promise problems are restricted to the ones over unary alphabets, we obtain the subclass $\mathcal{C}/\mathrm{unary}$. Those exotic notations come from [8, 9] and are adopted in [11, 16, 17, 18, 19, 21].

A family $\mathcal{M} = \{M_n\}_{n\in\mathbb{N}}$ of 2nfa's (resp., 2npda's) over alphabet Σ is of *polynomial size* if there exists a polynomial p satisfying $sc(M_n) \leq p(n)$ (resp., $ssc(M_n) \leq p(n)$) for all $n \in \mathbb{N}$. Similar notions are definable for other machine models, such as 2dfa's, 2ncta's, 2npda's, 2npdcta's, and 2dpdcta's.

Given a family $\mathcal{L} = \{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ of promise problems, a family $\mathcal{M} = \{M_n\}_{n\in\mathbb{N}}$ of nondeterministic machines, and a polynomial p, we say that M_n *solves* $(L_n^{(+)}, L_n^{(-)})$ *within time* $p(n, |x|)$ if (1) for any $x \in L_n^{(+)}$, there exists an accepting computation path of M_n on x having length at most $p(n, |x|)$ and (2)

[4] An ε-move of an automaton refers to the case where the automaton makes a transition without reading any input symbol.

for any $x \in L_n^{(-)}$, there is no accepting computation path of M_n on x but there is a rejecting computation path of having length at most $p(n,|x|)$. Moreover, $\mathcal{M}$ is said to *solve $\mathcal{L}$ in polynomial time* if there is a polynomial p such that, for all indices $n \in \mathbb{N}$, M_n solves $(L_n^{(+)}, L_n^{(-)})$ within time $p(n,|x|)$. We define 2N as the collection of all families of promise problems solvable by nonuniform families of polynomial-size 2nfa's in polynomial time.[5] By replacing these 2nfa's with 2dfa's, 2dpda's, and 2npda's, we respectively obtain 2D, 2DPD, and 2NPD. For any $k \in \mathbb{N}^+$, we further define $\mathrm{2NCT}_k$ and $\mathrm{2NPDCT}_k$ using k-counter 2ncta's and k-counter 2npdcta's,[6] respectively. When $k=1$, we tend to drop the subscript "k" and write, e.g., 2NCT instead of $\mathrm{2NCT}_1$. The notation co-$\mathcal{L}$ denotes $\{(L_n^{(-)}, L_n^{(+)})\}_{n\in\mathbb{N}}$. Given a complexity class $\mathcal{C}$ of promise problem families, such as 2N and 2NPD, co-$\mathcal{C}$ expresses the class $\{\text{co-}\mathcal{L} \mid \mathcal{L} \in \mathcal{C}\}$. It follows that 2D = co-2D and 2DPD = co-2DPD.

In addition, the use of unambiguous 1nfa's and unambiguous 2npda's introduces the complexity classes 2U and 2UPD, respectively, and their multi-counter variants $\mathrm{2UCT}_k$ and $\mathrm{2UPDCT}_k$. It then follows that $\mathrm{2D} \subseteq \mathrm{2DPD} \subseteq \mathrm{2UPD} \subseteq \mathrm{2UPDCT}$ and $\mathrm{2U} \subseteq \mathrm{2N} \subseteq \mathrm{2NCT} \subseteq \mathrm{2NPD} \subseteq \mathrm{2NPDCT}$.

2.3 Reducing the Number of Counters in Use

It is possible to reduce the number of counters in use on multi-counter automata and multi-counter pushdown automata. Minsky [12] earlier demonstrated how to simulate a Turing machine on a 2-counter automaton with exponential overhead. Since we cannot use the same simulation technique due to its large overhead, we need to take another, more direct approach toward $\mathrm{2NCT}_k$ and $\mathrm{2NPDCT}_k$. Even without the requirement of polynomial ceiling, it is possible in general to reduce the number of counters in use down to "4" for 2ncta's and "3" for 2npdcta's as shown below.

Proposition *2.1* *For any constants $k, k' \in \mathbb{N}$ with $k \geq 4$ and $k' \geq 3$, $\mathrm{2NCT}_k = \mathrm{2NCT}_4$ and $\mathrm{2NPDCT}_{k'} = \mathrm{2NPDCT}_3$. The same holds for the deterministic case.*

A core of the proof of Proposition 2.1 is the following lemma on the simulation of every pair of counters by a single counter with the heavy use of an appropriately defined "pairing" function. Let $\mathcal{M} = \{M_n\}_{n\in\mathbb{N}}$ denote any polynomial-size family of 2ncta's or of 2npdcta's running in time, in particular, $(n|x|)^t$ for a fixed constant $t \in \mathbb{N}^+$. This $\mathcal{M}$ satisfies the following lemma.

Lemma 2.2 *There exists a fixed deterministic procedure by which any single move of push/pop operations of two counters of M_n can be simulated by a series of operations with one counter with the help of 3 extra counters. These extra 3 counters are emptied after each simulation and thus they are reusable for any other purposes. If we freely use a stack during this simulation procedure, then we need only two extra counters instead of three. The state complexity of the procedure is $n^{O(1)}$.*

Proof of Proposition 2.1. We first look into the case of $\mathrm{2NCT}_k$ for every $k \geq 4$ and we wish to prove in the following that $\mathrm{2NCT}_k \subseteq \mathrm{2NCT}_4$.

Let $\mathcal{M} = \{M_n\}_{n\in\mathbb{N}}$ denote any nonuniform family of polynomial-size k-counter 2ncta's running in polynomial time, where M_n has the form $(Q_n, \Sigma, k, \{1, \bot\}, \delta_n, q_{0,n}, \bot, Q_{acc,n}, Q_{rej,n})$. Take two polynomials p_1 and p_2 such that $|Q_n| \leq p_1(n)$ and M_n halts within time $p_2(n,|x|)$ for any $n \in \mathbb{N}$ and any $x \in \Sigma^*$.

In what follows, we fix n and x arbitrarily. We abbreviate $p_2(n, p_1(n)) + 1$ as p and consider the pairing function $\langle i_1, i_2 \rangle_p$ defined as $\langle i_1, i_2 \rangle_p = i_1 \cdot p + i_2$ for any $i_1, i_2 \in [0, p-1]_{\mathbb{Z}}$.

We first group together all counters into pairs and apply Lemma 2.2 to simulate each pair of counters by a single counter with the help of three extra reusable counters, say, CT1–CT3. We repeat this process until there remains one counter other than CT1–CT3. Since there are only four counters left unremoved, this shows that $\mathrm{2NCT}_k \subseteq \mathrm{2NCT}_4$.

Next, we intend to show that $\mathrm{2NPDCT}_{k'} \subseteq \mathrm{2NPDCT}_3$ for any $k' \geq 3$. In this case, we follow the same argument as described above by removing counters except for CT1–CT3. Finally, four counters are

[5] As shown by Geffert et al. [4], in the case of 2dfa's and 2nfa's, removing the "polynomial time" requirement does not change the definitions of 2D and 2N.

[6] We remark that, with the use of multiple counters, it is possible to force 2ncta's and 2npdcta's to halt within polynomial time on *all computation paths*.

left unremoved. As shown in Lemma 2.2, we can further reduce the number of counters to three. This is because we can utilize a stack, which is originally provided by an underlying pushdown automaton. □

It is not clear that "4" and "3" are the smallest numbers supporting Proposition 2.1 for 2ncta's and 2npdcta's, respectively. We may conjecture that $2\mathrm{NCT}_i \neq 2\mathrm{NCT}_{i+1}$ and $2\mathrm{NPDCT}_j \neq 2\mathrm{NPDCT}_{j+1}$ for all $i \in [3]$ and $j \in [2]$.

3 Reachability with No Polynomial Ceiling Bounds

Let us consider the question raised in Section 1.1 on the closure property under complementation, namely, the 2N =?co-2N question. Unfortunately, we do not know its answer. With the presence of "counters", however, it is possible to provide a complete solution to this question.

Theorem 3.1 *For any constant $k \geq 4$,* co-$2\mathrm{NCT}_k \subseteq 2\mathrm{NCT}_4$.

Corollary 3.2 $2\mathrm{NCT}_4 =$ co-$2\mathrm{NCT}_4$.

A key to the proof of Theorem 3.1 is the following lemma.

Lemma 3.3 *For any constant $k \in \mathbb{N}^+$,* co-$2\mathrm{NCT}_k \subseteq 2\mathrm{NCT}_{5k+12}$.

The proof of Theorem 3.1 is described as follows. By Proposition 2.1, it suffices to consider the case of $k = 4$. We then obtain co-$2\mathrm{NCT}_4 \subseteq 2\mathrm{NCT}_{32}$ by Lemma 3.3. Proposition 2.1 again leads to $2\mathrm{NCT}_{32} \subseteq 2\mathrm{NCT}_4$. Therefore, co-$2\mathrm{NCT}_4 \subseteq 2\mathrm{NCT}_4$ follows. By taking the "complementation" of the both sides of this inclusion, we also obtain $2\mathrm{NCT}_4 \subseteq$ co-$2\mathrm{NCT}_4$. Corollary 3.2 is thus obtained.

To prove Lemma 3.3, we wish to use an algorithmic technique known as *inductive counting*. This intriguing technique was discovered independently by Immerman [6] and Szelepcsényi [15] in order to prove that NL = co-NL.

For the description of the proof of Lemma 3.3, we introduce the following special notion of (internal) configurations for a family $\{M_n\}_{n\in\mathbb{N}}$ of the nth k-counter 2ncta M_n running in time polynomial, say, $r(n,|x|)$. A *configuration* of M_n on input x is of the form $(q,l,\vec{m})$ with $q \in Q$, $l \in [0,|x|+1]_{\mathbb{Z}}$, and $\vec{m} = (m_1, m_2, \ldots, m_k) \in \mathbb{N}^k$. This form indicates that M_n is in inner state q, scanning the lth tape cell, and M_n's ith counter holds a number m_i for each index $i \in [k]$. We abbreviate as $CONF_{n,|x|}$ the configuration space $Q_n \times [0,|x|+1]_{\mathbb{Z}} \times [0,r(n,|x|)]_{\mathbb{Z}}^k$. Given two configurations c_1 and c_2 of M_n on x, the notation $c_1 \vdash_x c_2$ means that c_2 is "reachable" from c_1 by making a single move of M_n on x. We abbreviate as $c_1 \vdash_x^{t-1} c_t$ a chain of transitions $c_1 \vdash_x c_2 \vdash_x \cdots \vdash_x c_t$. Moreover, $\vdash_x^*$ denotes the transitive closure of $\vdash_x$.

Proof of Lemma 3.3. Let $k \in \mathbb{N}^+$ and let $\mathcal{L} = \{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ be any family in co-$2\mathrm{NCT}_k$. Since co-$\mathcal{L} \in 2\mathrm{NCT}_k$, there is a nonuniform family $\{M_n\}_{n\in\mathbb{N}}$ of polynomial-size k-counter 2ncta's that solves co-$\mathcal{L}$ in polynomial time. Since the runtime of M_n is polynomially bounded for all valid instances x, there is a polynomial r such that the value $r(n,|x|)$ upper-bounds the runtime of M_n on valid input x for any index $n \in \mathbb{N}$. Note that each counter must hold only a number between 0 and $r(n,|x|)$.

Our goal is to build a $(5k+12)$-counter 2ncta P_n, which solves the promise problem $(L_n^{(+)}, L_n^{(-)})$ for each index $n \in \mathbb{N}$. A basic idea of constructing such a machine P_n is to provide a procedure of deciding nondeterministically whether M_n rejects input x; in other words, whether all computation paths of M_n on x reach non-accepting inner states. For this purpose, we need to "count" the number of rejecting computation paths of M_n on x. If this number matches the total number of computation paths, then we know that M_n rejects x. From this follows $\mathcal{L} \in 2\mathrm{NCT}_{5k+12}$.

We arbitrarily fix a number n and a valid input x. In what follows, we deal with configurations of the form $(q,l,\vec{m})$ in $CONF_{n,|x|}$. It is important to note that, with the use of additional $k+1$ counters, we can "enumerate" all elements in $CONF_{n,|x|}$, ensuring a linear order on $CONF_{n,|x|}$. This fact makes it possible for us to select the elements of $CONF_{n,|x|}$ sequentially one by one in the following construction of P_n. For each number $i \in [0,r(n,|x|)]_{\mathbb{Z}}$, we define $V_i = \{(q,l,\vec{m}) \in CONF_{n,|x|} \mid (q_0,0,\vec{0}) \vdash_x^i (q,l,\vec{m})\}$ and set $N_i = |V_i|$. We wish to calculate the value $N_{r(n,|x|)}$ by inductively calculating N_i for each $i \in [0,r(n,|x|)]_{\mathbb{Z}}$ using additional counters.

Hereafter, we intend to calculate N_i inductively in the following fashion. Let i denote an arbitrary number in $[0, r(n, |x|)]_{\mathbb{Z}}$. We need another counter, say, CT1 to remember the value i. Since $0 \leq N_i \leq |CONF_{n,|x|}| = |Q_n|(r(n, |x|)+1)^k(|x|+2)$, we need to remember the value N_i using an additional counter, say, CT2. When $i = 0$, N_0 clearly equals 1. Assume that $1 \leq i \leq r(n, |x|)$. We use two parameters c and d, ranging over $[0, |CONF_{n,|x|}|]_{\mathbb{Z}}$, whose values are stored into two extra counters, say, CT3 and CT4. Furthermore, we need to remember the current location of M_n's tape head using another counter, say, CT5. To hold $\vec{m}$, we only need extra k counters, called eCT1–eCTk. During the following inductive procedure, we must empty the additionally introduced counters and reuse them to avoid a continuous introduction of new counters.

Initially, we set $c = 0$ in CT3. In a sequential way described above, we pick the elements $(q, l, \vec{m})$ from $CONF_{n,|x|}$ one by one. We store $(q, l, \vec{m})$ into CT5 and eCT1–eCTk, where q is remembered in the form of inner states. For each element $(q, l, \vec{m})$, we select nondeterministically either the process (1) or the process (2) described below, and execute it. After all elements $(q, l, \vec{m})$ are selected sequentially and either (1) or (2) is executed properly, we define $\hat{N}_i$ to be the current value of c (by removing the content of CT3 into CT2 to empty CT3).

To utilize the stored values of i, N_i, l, and $\vec{m}$, however, we need to copy them into extra $k+3$ counters, say, CT1$'$, CT2$'$, CT5$'$, and eCT1$'$–eCTk' (by bypassing another extra counter, say, CT10) and use these copied counters in the following process.

(1) Choose nondeterministically a computation path, say, γ of M_n on x and check whether $(q_0, 0, \vec{0}) \vdash_x^i (q, l, \vec{m})$ is true on this path γ. This is done by first returning a tape head to the start cell (i.e., the leftmost tape cell), preparing additional $k + 1$ counters, say, CT5$''$ and eCT1$''$–eCTk'', emptying them, and then simulating M_n on x for the first i steps using CT1$'$ and these new counters. Let $(p, h, \vec{s})$ denote the configuration reached after i steps from $(q_0, 0, \vec{0})$. We then compare between $(p, h, \vec{m})$ and $(q, l, \vec{m})$ by simultaneously decrementing the corresponding counters CT5$'$, eCT1$'$–eCTk', CT5$''$, and eCT1$''$–eCTk''. If the comparison is successful, then we increment the value of c by one. Otherwise, we reject x and halt.

(2) Initially, we set $d = 0$ in CT4. In the aforementioned sequential way, we pick the elements $(p, h, \vec{s})$ from $CONF_{n,|x|}$ one by one. For each selected element $(p, h, \vec{s})$, we need to store $(p, h, \vec{s})$ in other $k + 1$ counters and copy their contents into the existing $k + 1$ counters (as in (1)). For the simulation of M_n, to avoid introducing more counters, we reuse CT5$''$ and eCT1$''$–eCTk''. Follow nondeterministically a computation path, say, ξ and check whether $(q_0, 0, \vec{0}) \vdash_x^{i-1} (p, h, \vec{s})$ is true on ξ. As in (1), this is done by the use of the counters. If this is true, then we increment d by one. We then check whether $(p, h, \vec{s}) \vdash_x (q, l, \vec{m})$. If so, reject x and halt. After all elements $(p, h, \vec{s})$ are properly processed without halting, we check whether d matches $\hat{N}_{i-1}$. If not, reject x and halt.

Assume that the above inductive procedure ends after i reaches $r(n, |x|)$. We sequentially pick all elements $(z, t, \vec{e})$ from $(Q_n - Q_{acc,n}) \times [0, |x| + 1]_{\mathbb{Z}} \times [0, r(n, |x|)]_{\mathbb{Z}}^k$ one by one, store one into CT5 and eCT1–eCTk, and conduct the same procedure as described above, except for the following point. At the end of the procedure, instead of defining $\hat{N}_{r(n,|x|)+1}$, we check whether c equals $\hat{N}_{r(n,|x|)}$. If so, then we accept x; otherwise, we reject x.

It is possible to prove that, in a certain computation path of P_n, the value $\hat{N}_i$ correctly represents N_i for any index $i \in [0, r(n, |x|)]_{\mathbb{Z}}$. The correctness of the value of $\hat{N}_i$ can be proven by induction on i as in [6]. Therefore, P_n correctly solves $(L_n^{(+)}, L_n^{(-)})$. The above procedure can be implemented on an appropriate 2ncta with the total of $5(k + 1) + 7$ counters. Thus, $\mathcal{L}$ belongs to 2NCT$_{5k+12}$. $\square$

4 Effects of Polynomial Ceiling Bounds

We will see significant effects given by the restriction onto polynomially long input strings.

4.1 Elimination of Counters

In Section 2.3, we have discussed how to reduce the number of counters down to three or four. In the presence of polynomial ceilings, it is further possible to eliminate all counters from counter automata and counter pushdown automata.

Theorem 4.1 *Let k be an arbitrary constant in $\mathbb{N}^+$. (1)* $2\mathrm{NCT}_k/\mathrm{poly} = 2\mathrm{N}/\mathrm{poly}$. *(2)* $2\mathrm{NPDCT}_k/\mathrm{poly} = 2\mathrm{NPD}/\mathrm{poly}$. *The same statement holds even if underlying nondeterministic machines are changed to deterministic ones.*

Proof. Here, we prove only (2) since the proof of (1) is in essence similar. Since $2\mathrm{NPD} \subseteq 2\mathrm{NPDCT}_k$ for all $k \geq 1$, we obtain $2\mathrm{NPD}/\mathrm{poly} \subseteq 2\mathrm{NPDCT}_k/\mathrm{poly}$. We next show that $2\mathrm{NPDCT}_k/\mathrm{poly} \subseteq 2\mathrm{NPD}/\mathrm{poly}$. Consider any family $\mathcal{L} = \{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ of promise problems in $2\mathrm{NPDCT}_k/\mathrm{poly}$ over alphabet Σ. There exists a polynomial p satisfying $L_n^{(+)} \cup L_n^{(-)} \subseteq \Sigma^{\leq p(n)}$ for all $n \in \mathbb{N}$. Take any family $\{M_n\}_{n\in\mathbb{N}}$ of polynomial-size 2npdcta's that solves $\mathcal{L}$ with k counters in time polynomial in $(n, |x|)$. Let q denote a polynomial such that, for any $n \in \mathbb{N}$ and any promised input $x \in \Sigma^*$, $q(n, |x|)$ upper-bounds the runtime of M_n on x. Since p is a ceiling, all the k counters of M_n hold the number at most $q(n, p(n))$ on all "valid" instances. Define $r(n) = q(n, p(n))$. Since r is a polynomial, it is possible to express the contents of the k counters in the form of inner states. Therefore, without using any counter, we can simulate M_n on the input x by running an appropriate 2npda whose stack-state complexity is polynomially bounded. This implies that $\mathcal{L} \in 2\mathrm{NPD}/\mathrm{poly}$.

The last part of the theorem follows in a similar way as described above. □

Corollary 4.2 *(1) If* $2\mathrm{DCT}_4 = 2\mathrm{NCT}_4$*, then* $\mathrm{L}/\mathrm{poly} = \mathrm{NL}/\mathrm{poly}$. *(2) If* $2\mathrm{DPDCT}_3 = 2\mathrm{NPDCT}_3$*, then* $\mathrm{LOGDCFL}/\mathrm{poly} = \mathrm{LOGCFL}/\mathrm{poly}$.

The proof of this corollary is obtainable from Theorem 4.1 as well as the following two results. (i) $2\mathrm{D}/\mathrm{poly} = 2\mathrm{N}/\mathrm{poly}$ implies $\mathrm{L}/\mathrm{poly} = \mathrm{NL}/\mathrm{poly}$ [10]. (ii) $2\mathrm{DPD}/\mathrm{poly} = 2\mathrm{NPD}/\mathrm{poly}$ implies $\mathrm{LOGDCFL}/\mathrm{poly} = \mathrm{LOGCFL}/\mathrm{poly}$ [17].

4.2 Unambiguity of 2N and 2NPD

We turn our attention to the question of whether we can make 2nfa's and 2npda's unambiguous. In the simple case of unary inputs, let us recall the result of Geffert and Pighizzini [5], who showed that any 2nfa can be simulated by an appropriate 2ufa with a polynomial increase of the state complexity of the 2nfa. From this follows the collapse of 2N/unary to 2U/unary as stated in Fig. 1.

Next, we look into a relationship between 2NPD and 2UPD when all promise problems are limited to having polynomial ceilings.

Theorem 4.3 *(1)* $2\mathrm{N}/\mathrm{poly} = 2\mathrm{U}/\mathrm{poly}$. *(2)* $2\mathrm{NPD}/\mathrm{poly} = 2\mathrm{UPD}/\mathrm{poly}$.

The statement (1) of Theorem 4.3 was already proven in [19]. However, we do not know whether $2\mathrm{N} = 2\mathrm{U}$ or $2\mathrm{NCT}_k = 2\mathrm{UCT}_k$ even though (1) holds.

Hereafter, we focus on proving the statement (2) of Theorem 4.3. For this purpose, we use the nonuniform (i.e., the Karp-Lipton style polynomial-size advice-enhanced extension) computational model of *two-way auxiliary unambiguous pushdown automata* (or aux-2upda's), which are 2upda equipped with auxiliary work tapes, and the complexity class $\mathrm{UAuxPDA{,}TISP}(n^{O(1)}, \log n)/\mathrm{poly}$ induced by aux-2upda's[7] that run in time $n^{O(1)}$ using work space $O(\log n)$ together with polynomially-bounded advice functions. Reinhardt and Allender [13] demonstrated that LOGCFL/poly coincides with $\mathrm{UAuxPDA{,}TISP}(n^{O(1)}, \log n)/\mathrm{poly}$.

Discovered in [16] was a close connection between parameterized decision problems and families of promise problems solvable by certain finite automata families. We quickly review necessary terminology, introduced in [20] and fully developed in [16, 17, 18, 19, 21]. A *parameterized decision problem* over alphabet Σ is of the form (L, m), where $L \subseteq \Sigma^*$ and $m(\cdot)$ is a size parameter (i.e., a mapping of Σ^* to $\mathbb{N}$). Any size parameter computable by an appropriate log-space deterministic Turing machine (DTM) is called a *logspace size parameter*. A typical example is $m_{\|}$ defined as $m_{\|}(x) = |x|$ for all $x \in \Sigma^*$. All parameterized decision problems whose size parameters m satisfy $|x| \leq q(m(x))$ for all $x \in \Sigma^*$ form the complexity class PHSP, where q is an appropriate polynomial depending only on m. See [18] for more information. The notation para-LOGCFL/poly denotes the collection of all parameterized decision problems (L, m) with logspace size parameters m solvable by *two-way auxiliary nondeterministic*

[7] This means aux-2upda's running in $n^{O(1)}$ time using $O(\log n)$ work space on any promised instance of the form $(x, h(|x|))$, where h is a polynomially-bounded advice function.

pushdown automata (or aux-2npda's) running in time $m(x)^{O(1)}$ and space $O(\log m(x))$ with the use of advice strings of length polynomial in $m(x)$, where x represents an "arbitrary" input. As a special case, if m is fixed to $m_{\|}$, we obtain LOGCFL/poly.

Let $\mathcal{L} = \{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ and $\mathcal{K} = \{(\mathcal{K}_n^{(+)}, \mathcal{K}_n^{(-)})\}_{n\in\mathbb{N}}$ be any two families of promise problems over a common alphabet Σ. Given a parameterized decision problem (L, m) over Σ, we say that $\mathcal{L}$ is *induced from* (L, m) if, for any index $n \in \mathbb{N}$, $L_n^{(+)} = L \cap \Sigma_{(n)}$ and $L_n^{(-)} = \overline{L} \cap \Sigma_{(n)}$, where $\Sigma_{(n)} = \{x \in \Sigma^* \mid m(x) = n\}$. The family $\mathcal{K}$ is said to be an *extension* of $\mathcal{L}$ if $L_n^{(+)} \subseteq K_n^{(+)}$ and $L_n^{(-)} \subseteq K_n^{(-)}$ hold for any $n \in \mathbb{N}$. Moreover, $\mathcal{L}$ is *L-good* if the set $\{1^n\#x \mid n \in \mathbb{N}, x \in L_n^{(+)} \cup L_n^{(-)}\}$ belongs to L. A collection $\mathcal{F}$ of promise problem families is *L-good* if all elements of $\mathcal{F}$ has an L-good extension in $\mathcal{F}$. From $\mathcal{L}$, we define $K_n^{(+)} = \{1^n\#x \mid x \in L_n^{(+)}\}$ and $K_n^{(-)} = \{1^n\#x \mid x \notin L_n^{(+)}\} \cup S_n$ for any $n \in \mathbb{N}$, where $\Sigma_{\#} = \Sigma \cup \{\#\}$ and $S_n = \{z\#x \mid z \in \Sigma^n - \{1^n\}, x \in (\Sigma_{\#})^*\} \cup \Sigma^n$. Finally, we say that (K, m) is *induced from* $\mathcal{L}$ if $K = \bigcup_{n\in\mathbb{N}} K_n^{(+)}$ and $m : (\Sigma_{\#})^* \to \mathbb{N}$ satisfies that (i) $m(w) = n$ holds for any string w of the form $1^n\#x$ with $x \in L_n^{(+)} \cup L_n^{(-)}$ and (ii) $m(w) = |w|$ holds for all other strings w. Note that $\overline{K}$ equals $\bigcup_{n\in\mathbb{N}} K_n^{(-)}$. See [18] for more information. The L-goodness can be proven for 2NPD/poly, 2UPD/poly, and co-2NPD/poly in a way similar to [17]. Since the set $\{1^n\#x \mid x \in \mathbb{N}, x \in L_n^{(+)} \cup L_n^{(-)}\}$ does not alter by exchanging between $L_n^{(+)}$ and $L_n^{(-)}$, the L-goodness of co-2NPD/poly follows immediately.

In the rest of this work, we succinctly write LOGUCFL/poly for UAuxPDA,TISP($n^{O(1)}, \log n$)/poly. Notice that LOGCFL is characterized in terms of polynomial-time log-space aux-2npda's. This characterization holds even under the presence of advice.

Proof of Theorem 4.3. The statement (1) was already proven in [19]. Thus, we focus on the statement (2) and provide its proof. Since unambiguity is a special case of nondeterminism, 2UPD is obviously contained in 2NPD. In what follows, we intend to prove that 2NPD/poly $\subseteq$ 2UPD because this statement is logically equivalent to 2NPD/poly = 2UPD/poly.

To avoid a cumbersome repetition of the description of "parameterized decision problems" and "logspace size parameters" given in pages 9–10, we skip the whole description here.

In the rest of this proof, we succinctly write LOGUCFL/poly for UAuxPDA,TISP($n^{O(1)}, \log n$)/poly. Let us assert the following claim concerning LOGUCFL/poly and 2UPD. A similar claim was proven first in [18], and later proven in [19] for UL/poly and in [17] for LOGCFL/poly. Refer to these references for more precise information.

Claim 4.4 *(i)* LOGCFL $\subseteq$ LOGUCFL/poly *implies* para-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGUCFL/poly. *(ii)* para-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGUCFL/poly *implies* 2NPD/poly $\subseteq$ 2UPD.

It is known that LOGCFL/poly $\subseteq$ LOGUCFL/poly [13]. From this fact, Claim 4.4 therefore concludes that 2NPD/poly $\subseteq$ 2UPD, as requested.

Hereafter, we wish to give the proof of Claim 4.4. Let us begin with the proof of Claim 4.4(i). Assume that LOGCFL $\subseteq$ LOGUCFL/poly and let (L, m) denote any parameterized decision problem in para-LOGCFL/poly $\cap$ PHSP. Since m is polynomially honest, there is a polynomial q satisfying $|x| \leq q(m(x))$ for all x. We intend to turn (L, m) into its corresponding language, say, P as follows. Since $(L, m) \in$ para-LOGCFL/poly, there are a polynomially-bounded advice function h and an aux-2npda M_0 such that M_0 recognizes the language $\{(x, h(m(x))) \mid x \in L\}$ in time $m(x)^{O(1)}$ and space $O(\log m(x))$. We define $L_n^{(+)} = L \cap \Sigma_{(n)}$ and $L_n^{(-)} = \overline{L} \cap \Sigma_{(n)}$, where $\Sigma_{(n)} = \{x \in \Sigma^* \mid m(x) = n\}$. The desired set P is defined by $P = \{(x, 1^t) \mid x \in L, t \in \mathbb{N}, m(x) \leq t\}$. To show that $P \in$ LOGCFL/poly, it suffices to consider the following algorithm: on input $(x, 1^t)$, first check whether $m(x) \leq t$ using log space. Otherwise, reject the input immediately. We then run M_0 on $(x, h(m(x)))$ by the help of advice string, which must have the form $h(0)\#h(1)\#\cdots\#h(t)$. Our assumption then guarantees that P belongs to LOGUCFL/poly. From this P, it is possible to define an advice function g and an aux-2npda N for which L is recognized by N in time $m(x)^{O(1)}$ using space $O(\log m(x))$ with the help of g. Hence, we can conclude that (L, m) falls in para-LOGUCFL/poly.

We then prove Claim 4.4(ii). A key to its proof is the following two statements (a)–(b), which are analogous to [18, Proposition 5.1]. Let $\mathcal{L}$ denote any family $\{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ of promise problems, let L and K be any two decision problems (or equivalently, languages) over the common alphabet Σ, and let m denote any logspace size parameter over Σ. We write $(\mathcal{C},\mathcal{D})$ for any element of the set $\{(\mathrm{LOGCFL}, \mathrm{2NPD}), (\mathrm{LOGUCFL}, \mathrm{2UPD})\}$.

(a) If $\mathcal{L}$ is induced from (L,m), then $(L,m) \in$ para-$\mathcal{C}$/poly $\cap$ PHSP iff $\mathcal{L} \in \mathcal{D}$/poly.

(b) If $\mathcal{L}$ is L-good and (K,m) is induced from $\mathcal{L}$, then $(K,m) \in$ para-$\mathcal{C}$/poly $\cap$ PHSP iff $\mathcal{L} \in \mathcal{D}$/poly.

Note that the validity of the above statements (a)–(b) for the first case of $\mathcal{C}$ = LOGCFL and $\mathcal{D}$ = 2NPD comes directly from [17, Proposition 4.3]. The second case of $\mathcal{C}$ = LOGUCFL and $\mathcal{D}$ = 2UPD will be handled later.

Meanwhile, we set our goal to deriving 2NPD/poly $\subseteq$ 2UPD from the inclusion para-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGUCFL/poly. Take any family $\mathcal{L} = \{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ of promise problems in 2NPD/poly over alphabet Σ and take the parameterized decision problem (K,m) induced from $\mathcal{L}$. As remarked earlier, all elements in 2NPD are L-good. By (b), it thus follows that $(K,m) \in$ para-LOGCFL/poly $\cap$ PHSP iff $\mathcal{L} \in$ 2NPD/poly. Hence, we obtain $(K,m) \in$ para-LOGCFL/poly. If para-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGUCFL/poly, then we conclude that $(K,m) \in$ para-LOGUCFL/poly. By (b) again, $\mathcal{L}$ must belong to 2UPD/poly. Therefore, Claim 4.4(ii) is true.

The remaining task is to prove the statement (a)–(b) for the second case of $\mathcal{C}$ = LOGUCFL and $\mathcal{D}$ = 2UPD. Following an argument of [17, 18, 19], we first show (a). Assume that $\mathcal{L}$ is induced from (L,m) and that (L,m) is in para-LOGUCFL/poly$\cap$PHSP. Take a polynomially-bounded advice function h and an aux-2upda M that recognizes L in polynomial time using log space with the help of the advice string $h(m(x))$ given according to each input x. Since $L_n^{(+)} = L \cap \Sigma_{(n)}$ and $L_n^{(-)} = \overline{L} \cap \Sigma_{(n)}$ (where $\Sigma_{(n)} = \{x \mid m(x) = n\}$), it is possible to convert M with h working on all inputs x in $\Sigma_{(n)}$ into an appropriate 2nfa N_n by integrating h into M and simulating it on the 2nfa N_n using polynomially many inner states. A detailed description is given as a supporting lemma in [18, Lemma 5.3] for a conversion of a single (quantum) Turing machine into another two-way (quantum) finite automaton. Notice that N_n is unambiguous since so is M with h. Thus, $\mathcal{L}$ belongs to 2UPD.

On the contrary, assume that $\mathcal{L} \in$ 2UPD/poly. Take a family $\mathcal{M} = \{M_n\}_{n\in\mathbb{N}}$ of polynomial-size 2upda's solving $\mathcal{L}$ in polynomial time. We wish to convert $\mathcal{M}$ into a single polynomial-time log-space aux-2upda N solving (L,m) with a polynomially-bounded advice function h. A detailed description is given in [18, Lemma 5.2] for a conversion of a two-way (quantum) finite automata family into a (quantum) Turing machine as a supporting lemma.

Next, we intend to prove (b). Assume that $\mathcal{L}$ is L-good and (K,m) is induced from $\mathcal{L}$. If (K,m) is in para-LOGUCFL/poly $\cap$ PHSP, then there are a polynomially-bounded advice function h and an aux-2upda M that solves (K,m) in $O(m(z)^k)$ time using $O(\log m(z))$ space on every input z of the form $(x, h(m(x)))$. Let n denote an arbitrary number in $\mathbb{N}$. In a way similar to a conversion of an advised (quantum) Turing machine into a family of two-way (quantum) finite automata in the proof of [18, Lemma 5.5], we introduce the following 2nfa N_n.

We first encode a content of the work tape into an inner state and approximately define a new transition function of N_n so that it simulates M step by step. Consider arbitrary n and $x \in \Sigma_{(n)}$. Since m and h are polynomially bounded, the space usage of N_n is upper bounded by $O(\log m(z)) \subseteq O(\log n)$ and the runtime of N_n is upper-bounded by $O(m(z)^k) \subseteq O(n^{k'})$ for an appropriate constant $k' > k$. By the definition of N_n, the family $\{N_n\}_{n\in\mathbb{N}}$ turns out to solve $\mathcal{L}$. We thus conclude that $\mathcal{L} \in$ 2U/poly.

On the contrary, assume that $\mathcal{L} \in$ 2UPD/poly. Take a family $\mathcal{M} = \{M_n\}_{n\in\mathbb{N}}$ of polynomial-size 2upda's that solves $\mathcal{L}$ in polynomial time. Since $K = \bigcup_{n\in\mathbb{N}} K_n^{(+)}$ with $K_n^{(+)} = \{1^n\#x \mid x \in L_n^{(+)}\}$, we can construct an aux-2npda N that takes an input of the form $1^n\#x$ and simulates M_n on x, where a code of M_n is given as an advice string to N. Since M_n is unambiguous, so is N_n. Given in [18] is a detailed description of a conversion of a (quantum) finite automata family into a (quantum) Turing machine. □

4.3 Complementation of 2N and 2NPD

Let us look into the question raised in Section 1.1 concerning the closure property under complementation. Geffert, Mereghetti, and Pighizzini [4] earlier demonstrated a simulation of a "complementary" 2nfa (i.e., a two-way finite automaton making co-nondeterministic moves) on unary inputs by another 2nfa with a polynomial increase of the state complexity. From this fact, we instantly conclude that co-2N/unary coincides with 2N/unary as stated in Fig. 1.

Hereafter, we discuss the complementation closures of 2N and 2NPD when all valid instances are limited to polynomially long strings.

Theorem 4.5 *(1)* 2N/poly = co-2N/poly. *(2)* 2NPD/poly = co-2NPD/poly

It is important to note that the statement (1) does not require the use of parameterized version of NL. This is rather a direct consequence of Theorem 3.1 (and Corollary 3.2) together with Theorem 4.1(1). This fact exemplifies the strength of the additional use of counters provided to underlying finite automata.

Proof of Theorem 4.5. (1) By Corollary 2.2, $\mathrm{2NCT}_4 = \text{co-}\mathrm{2NCT}_4$ follows. Thus, we conclude that $\mathrm{2NCT}_4/\mathrm{poly} = \text{co-}\mathrm{2NCT}_4/\mathrm{poly}$. By Theorem 4.1(2), we then obtain 2N/poly = co-2N/poly.

(2) The proof for LOGCFL = co-LOGCFL in [2] can be carried over to the advised setting, and thus we obtain LOGCFL/poly = co-LOGCFL/poly. In analogy to Claim 4.4, we intend to assert the following claim.

Claim 4.6 *(i)* co-LOGCFL $\subseteq$ LOGCFL/poly *implies* para-co-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGCFL/poly. *(ii)* para-co-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGCFL/poly *implies* co-2NPD/poly $\subseteq$ 2NPD.

Claim 4.6 clearly leads to 2NPD/poly = co-2NPD/poly from LOGCFL/poly = co-LOGCFL/poly.

The proof of Claim 4.6 is in essence similar to that of Claim 4.4. Let us prove Claim 4.6(i). Assume that co-LOGCFL $\subseteq$ LOGCFL/poly. It is easy to show that this implies co-LOGCFL/poly = LOGCFL/poly. Let (L, m) denote any parameterized decision problem in para-co-LOGCFL/poly$\cap$PHSP over alphabet Σ. Since $(\overline{L}, m)$ belongs to para-LOGCFL/poly, there are a polynomially-bounded advice function h and an advised aux-2npda M such that M recognizes L in time $m(x)^{O(1)}$ using space $O(\log m(x))$ with the help of h, where x is a "symbolic" input. We define $P = \{(x, 1^t) \mid x \in L, t \in \mathbb{N}, m(x) \leq t\}$. By following an argument similar to the one for Claim 4.4(i), we can conclude that $\overline{P} \in$ LOGCFL/poly. Hence, P belongs to co-LOGCFL/poly. By our assumption, P also belongs to LOGCFL/poly. Take another advised aux-2npda N that recognizes L in time $n^{O(1)}$ and space $O(\log |x|)$ with an appropriate polynomially-bounded advice function. Since m is polynomially honest, (L, m) must belong to para-LOGCFL/poly.

We next target Claim 4.6(ii). Recall the statements (a)–(b) introduced in the proof of Theorem 4.3(2). We claim that the same statements (a)–(b) holds for any element $(\mathcal{C}, \mathcal{D})$ of the set $\{(\mathrm{LOGCFL}, \mathrm{2NPD}), (\text{co-LOGCFL}, \text{co-2NPD})\}$. Let $\mathcal{L}$ denote any family $\{(L_n^{(+)}, L_n^{(-)})\}_{n\in\mathbb{N}}$ of promise problems, let L and K be any two decision problems over the common alphabet Σ, and let m denote any logspace size parameter over Σ.

(a) If $\mathcal{L}$ is induced from (L, m), then $(L, m) \in$ para-$\mathcal{C}$/poly $\cap$ PHSP iff $\mathcal{L} \in \mathcal{D}$/poly.

(b) If $\mathcal{L}$ is L-good and (K, m) is induced from $\mathcal{L}$, then $(K, m) \in$ para-$\mathcal{C}$/poly $\cap$ PHSP iff $\mathcal{L} \in \mathcal{D}$/poly.

With the use of the statements (a)–(b), Claim 4.6(ii) is proven as follows. Assume that para-co-LOGCFL/poly $\cap$ PHSP $\subseteq$ para-LOGCFL/poly. Take an arbitrary family $\mathcal{L}$ in co-2NPD/poly. This family induces the parameterized decision problem (K, m). Recall that co-2NPD/poly is L-good. There exists an L-good extension of $\mathcal{L}$. By the nature of L-goodness, the existence of such an extension implies that $\mathcal{L}$ itself is also L-good. It then follows by (b) with $\mathcal{C}$ = co-LOGCFL and $\mathcal{D}$ = co-2NPD that $(K, m) \in$ para-co-LOGCFL/poly $\cap$ PHSP iff $\mathcal{L} \in$ co-2NPD/poly. Since $\mathcal{L} \in$ co-2NPD/poly, we thus conclude that (K, m) is in para-co-LOGCFL/poly. Our assumption then yields the containment $(K, m) \in$ para-LOGCFL/poly. By (b) with $\mathcal{C}$ = LOGCFL and $\mathcal{D}$ = 2NPD, it follows that $(K, m) \in$ para-LOGCFL/poly $\cap$ PHSP iff $\mathcal{L} \in$ 2NPD/poly. Thus, $\mathcal{L}$ must belong to 2NPD/poly.

The remaining task is to show the validity of the statements (a)–(b) for any element $(\mathcal{C}, \mathcal{D})$ of the set $\{(\mathrm{LOGCFL}, \mathrm{2NPD}), (\text{co-LOGCFL}, \text{co-2NPD})\}$. The first case of $\mathcal{C}$ = LOGCFL and $\mathcal{D}$ = 2NPD comes from [17, Proposition 4.3]. We thus intend to prove the second case of $\mathcal{C}$ = co-LOGCFL and $\mathcal{D}$ = co-2NPD. We first show (a). Assume that $\mathcal{L}$ is induced from (L, m) in para-co-LOGCFL/poly $\cap$ PHSP. Since co-$\mathcal{L}$ is also induced from $(\overline{L}, m)$ and $(\overline{L}, m)$ belongs to para-LOGCFL/poly $\cap$ PHSP, it follows from the first case that $(\overline{L}, m) \in$ para-LOGCFL/poly $\cap$ PHSP iff co-$\mathcal{L} \in$ 2NPD/poly. By taking complementation, we obtain the desired relationship stated in (a).

Next, we wish to prove (b). Assume that $\mathcal{L}$ is L-good and (K, m) is induced from $\mathcal{L}$. By the nature of L-goodness, co-$\mathcal{L}$ is also L-good. We succinctly write $\Sigma_{(n)}$ for $L_n^{(+)} \cup L_n^{(-)}$ for any $n \in \mathbb{N}$. Since $\mathcal{L}$ is L-good, there is a polynomial-time log-space DTM N_0 that recognizes $\{1^n\#x \mid n \in \mathbb{N}, x \in \Sigma_{(n)}\}$. For any $n \in \mathbb{N}$, let $S_n = \{z\#x \mid z \in \Sigma^n - \{1^n\}, x \in (\Sigma_\#)^*\} \cup \Sigma^n$. Note that, by the definition of K, K has the form $\bigcup_{n\in\mathbb{N}} K_n^{(+)}$, where $K_n^{(+)} = \{1^n\#x \mid x \in L_n^{(+)}\}$ and $K_n^{(-)} = \{1^n\#x \mid x \notin L_n^{(+)}\} \cup S_n$. It then follows that $\overline{K} = \bigcup_{n\in\mathbb{N}} K_n^{(-)}$. Let (P, m) denote the parameterized decision problem induced from co-$\mathcal{L}$. Similarly to K, P equals $\bigcup_{n\in\mathbb{N}} P_n^{(+)}$ with $P_n^{(+)} = \{1^n\#x \mid x \in L_n^{(-)}\}$ and $P_n^{(-)} = \{1^n\#x \mid x \notin L_n^{(-)}\} \cup S_n$. By applying [17, Proposition 4.3] to co-$\mathcal{L}$, it follows that $(P, m) \in$ para-LOGCFL/poly $\cap$ PHSP iff co-$\mathcal{L} \in$ 2NPD/poly. By taking complementation, we conclude that $(\overline{P}, m) \in$ para-co-LOGCFL/poly $\cap$ PHSP iff $\mathcal{L} \in$ co-2NPD/poly. Therefore, it suffices to verify that (*) $(\overline{K}, m) \in$ para-LOGCFL/poly $\cap$ PHSP iff $(P, m) \in$ para-LOGCFL/poly $\cap$ PHSP, because this equivalence shows that $(\overline{K}, m) \in$ para-LOGCFL/poly $\cap$ PHSP iff co-$\mathcal{L} \in$ 2NPD/poly. Therefore, we conclude that $(K, m) \in$ para-co-LOGCFL/poly $\cap$ PHSP iff $\mathcal{L} \in$ co-2NPD/poly.

Hereafter, we intend to verify (*) by assume that m is polynomially honest. Assume that $(\overline{K}, m) \in$ para-LOGCFL/poly. It is not difficult to show that $(K_n^{(-)} - S_n) \cap \Sigma_{(n)} = P_n^{(+)} \cap \Sigma_{(n)}$ $(= \{1^n\#x \mid x \in \Sigma_{(n)} \cap L_n^{(-)}\})$ for any $n \in \mathbb{N}$. Therefore, if $x \in \bigcup_{n\in\mathbb{N}} \Sigma_{(n)}$, then it follows that $x \in \overline{K} - \bigcup_{n\in\mathbb{N}} S_n$ iff $x \in P$. Since $\bigcup_{n\in\mathbb{N}} S_n$ is in L, this equivalence relation leads to (*), as requested.

This completes the entire proof of the theorem. □

References

[1] P. Berman and A. Lingas. On complexity of regular languages in terms of finite automata. Report 304, Inst. Comp. Sci., Polish Academy of Science, Warsaw (1977)

[2] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. SIAM J. Comput. 18 (1989) 559–578.

[3] V. Geffert. An alternating hierarchy for finite automata. Theor. Comput. Sci. 445, 1–24 (2012).

[4] V. Geffert, C. Mereghetti, and G. Pighizzini. Complementing two-way finite automata. Inf. Comput. 205, 1173–1187 (2007)

[5] V. Geffert and G. Pighizzini. Two-way unary automata versus logarithmic space. Inform. Comput. 209, 1016–1025 (2011)

[6] N. Immerman. Nondeterministic space is closed under complement. SIAM J. Comput. 17 (1988) 935–938.

[7] J. Hartmanis and S. R. Mahaney. Languages simultaneously complete for one-way and two-way log-tape automata. SIAM J. Comput. 10, 383–390 (1981)

[8] C. A. Kapoutsis. Size complexity of two-way finite automata. In the Proc. of DLT 2009, LNCS, vol. 5583, pp. 47–66, Springer (2009)

[9] C. A. Kapoutsis. Minicomplexity. J. Automat. Lang. Combin. 17 (2012) 205–224.

[10] C. A. Kapoutsis. Two-way automata versus logarithmic space. Theory Comput. Syst. 55 (2014) 421–447.

[11] C. A. Kapoutsis and G. Pighizzini. Two-way automata characterizations of L/poly versus NL. Theory Comput. Syst. 56 (2015) 662–685.

[12] M. L. Minsky. Computation: Finite and Infinite Machines. Prentice-Hall, Englewood Cliff, NJ (1967)

[13] K. Reinhardt and E. Allender. Making nondeterminism unambiguous. SIAM J. Comput. 29, 1118–1181 (2000)

[14] W. J. Sakoda and M. Sipser. Nondeterminism and the size of two-way finite automata. In the Proc. of STOC 1978, pp. 275–286 (1978)

[15] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. Acta Inform. 26 (1988) 279–284.

[16] T. Yamakami. State complexity characterizations of parameterized degree-bounded graph connectivity, sub-linear space computation, and the linear space hypothesis. Theor. Comput. Sci. 798, 2–22 (2019)

[17] T. Yamakami. Parameterizations of logarithmic-space reductions, stack-state complexity of nonuniform families of pushdown automata, and a road to the LOGCFL⊆LOGDCFL/poly question. Available at arXiv:2108.12779, 2021.

[18] T. Yamakami. Nonuniform families of polynomial-size quantum finite automata and quantum logarithmic-space computation with polynomial-size advice. Inf. Comput. 286, article 104783 (2022)

[19] T. Yamakami. Unambiguity and fewness for nonuniform families of polynomial-size nondeterministic finite automata. In the Proc. of RP 2022, LNCS, vol. 13608, pp. 77–92, Springer (2022). A corrected and expanded version is available at arXiv:2311.09979.

[20] T. Yamakami. The 2CNF Boolean formula satsifiability problem and the linear space hypothesis. J. Comput. Syst. Sci. 136, 88–112 (2023)

[21] T. Yamakami. Power of counting by nonuniform families of polynomial-size finite automata. In the Proc. of FCT 2023, LNCS, vol. 14292, pp. 421–435, Springer (2023). A corrected and expanded version is available at arXiv:2310.18965.