Two-Way Automata versus Logarithmic Space

Christos A. Kapoutsis

LIAFA – Université Paris VII

Abstract. We strengthen previously known connections between the size complexity of two-way finite automata (2FAs) and the space complexity of Turing machines. We prove that

- every s-state 2NFA can be simulated on all poly(s)-long inputs by some poly(s)-state 2DFA if and only if $\mathsf{NL} \subseteq \mathsf{L/poly}$ and
- every s-state 2NFA can be simulated on all $2^{\mathrm{poly}(s)}$-long inputs by some poly(s)-state 2DFA if and only if $\mathsf{NLL} \subseteq \mathsf{LL/polylog}$.

Here, 2DFAs and 2NFAs are the deterministic and nondeterministic 2FAs, NL and $\mathsf{L/poly}$ are the standard space complexity classes, and NLL and $\mathsf{LL/polylog}$ are their counterparts for $O(\log\log n)$ space and poly($\log n$) bits of advice. Our arguments strengthen and extend an old theorem by Berman and Lingas and can be used to obtain variants of the above statements for other modes of computation or other combinations of bounds for the input length, the space usage, and the length of advice.

1 Introduction

The question whether nondeterministic computations can be more powerful than deterministic ones is central in complexity theory. Numerous instantiations have been studied, for a variety of computational models under a variety of resource restrictions. This article investigates the connection between two kinds of such instantiations, those for *Turing machines* (TMs) under *space* restrictions and those for *two-way finite automata* (2FAs) under *size* restrictions.

On the one hand, the question whether nondeterminism makes a difference in space-bounded TMs is among the oldest in complexity theory. Formally, it asks whether there is a bound f with $\mathsf{DSPACE}(f) \not\supseteq \mathsf{NSPACE}(f)$.[(1)] For $f(n) = n$, this appeared already in [11]. We know that every such bound must be $\Omega(\log\log n)$, since otherwise all languages involved are regular [17,6,1], and that such bounds exist in $\Theta(\log\log n) \cup \Omega(\log n)$ iff $\log\log n$ is already one [13,18]. Since computation in space $\Omega(\log n)$ is more natural than in space $o(\log n)$, research focused on the $\log n$ bound and on the corresponding conjecture that $\mathsf{L} \not\supseteq \mathsf{NL}$. An early observation [2] was that, even if $\mathsf{L} \not\supseteq \mathsf{NL}$, a deterministic TM (DTM) of space $O(\log n)$ may still be able to simulate a nondeterministic one if it is allowed nonuniform behavior. This led to the introduction of the class $\mathsf{L/poly}$ of languages recognizable in space $O(\log n)$ by DTMs accessing poly(n) $\equiv n^{O(1)}$ bits of nonuniform advice [10], and to the stronger conjecture that even $\mathsf{L/poly} \not\supseteq \mathsf{NL}$.

* Research funded by a Marie Curie Intra-European Fellowship (PIEF-GA-2009-253368) within the European Union Seventh Framework Programme (FP7/2007-2013).

Note that, for only $O(\log n)$ advice bits, the resulting conjecture $\mathsf{L}/\mathsf{log} \not\supseteq \mathsf{NL}$ is equivalent to $\mathsf{L} \not\supseteq \mathsf{NL}$ [10] and that the classes $\mathsf{DSPACE}(f)/2^{O(f)}$ for varying f have been studied in [8] under the names $\mathsf{NUDSPACE}(f)$.

On the other hand, the question whether nondeterminism makes a difference in size-bounded 2FAs was posed already in [14]. Formally, it asks whether there exist s-state 2NFAs that admit no equivalent 2DFA with $\mathrm{poly}(s)$ states. A robust theoretical framework around this question appeared in [12], with the introduction of the complexity classes 2D and 2N. The former consists of every family of languages $(L_h)_{h\geq 1}$ recognizable by a family of $\mathrm{poly}(h)$-state 2DFAs, while the latter is the corresponding class for 2NFAs. The original question is thus equivalent to whether $\mathsf{2D} \not\supseteq \mathsf{2N}$. We remark that the answer is known to be positive if the 2DFAs must obey certain restrictions [14,15,7,9]; and negative if the 2DFAs are allowed quasi-polynomially many states and the languages are unary [4]. For the general case, the answer has been conjectured to be positive [14,12].

The above two questions are connected via the lengths of the strings necessary to confirm the conjecture $\mathsf{2D} \not\supseteq \mathsf{2N}$. To explain this, let us consider a language family $\mathcal{L} = (L_h)_{h\geq 1}$ witnessing the conjecture. Then $\mathcal{L} \in \mathsf{2N}$ but $\mathcal{L} \notin \mathsf{2D}$. The latter means that every family $\mathcal{B} = (B_h)_{h\geq 1}$ of $\mathrm{poly}(h)$-state 2DFAs fails to recognize $\mathcal{L}$, in the sense that *at least one* B_h fails to recognize the respective L_h. This is equivalent to saying that every such $\mathcal{B}$ contains *infinitely many* failing B_h (because, if some $\mathcal{B}$ contains only finitely many of them, then replacing those with larger ones that succeed we restore correctness without hurting polynomiality). Of course, a B_h fails to recognize L_h iff it errs on at least one input x_h. Putting everything together, we see that $\mathcal{L} \notin \mathsf{2D}$ iff for every family $\mathcal{B}$ of $\mathrm{poly}(h)$-state 2DFAs there is a family of inputs $\mathcal{x} = (x_h)_{h\geq 1}$ such that B_h errs on x_h for infinitely many h. Intuitively, these inputs constitute *hard instances*.

Definition. *A family of* hard instances of $\mathcal{L}$ for $\mathcal{B}$ *is any* $(x_h)_{h\geq 1}$ *where there exist infinitely many* h *such that* B_h *accepts* $x_h \iff x_h \notin L_h$.

For each $\mathcal{B}$, we can always find hard instances of at most exponential length.

Lemma 0. *Let* $\mathcal{L} \in \mathsf{2N}$. *If* $\mathcal{L} \notin \mathsf{2D}$ *then for every family of* $\mathrm{poly}(h)$*-state* 2DFAs *there is a family of* $2^{\mathrm{poly}(h)}$*-long hard instances of* $\mathcal{L}$.

However, in the proof of this lemma,[(2)] the degree of the polynomial for the 2DFAs lower-bounds the degree of the polynomial exponent for the hard instances. Hence, as the 2DFAs become polynomially larger, the guaranteed hard instances become super-polynomially longer. We thus have no exponential *global* upper bound, for the lengths of hard instances over *all* families of $\mathrm{poly}(h)$-state 2DFAs.

Definition. *A* g-long witness for $\mathsf{2D} \not\supseteq \mathsf{2N}$ *is an* $\mathcal{L} \in \mathsf{2N}$ *such that for every family of* $\mathrm{poly}(h)$*-state* 2DFAs *there is a family of* $g(h)$*-long hard instances of* $\mathcal{L}$.

So, by Lemma 0, if $\mathsf{2D} \not\supseteq \mathsf{2N}$ then super-exponentially long witnesses (g-long, where $g = \Omega(g')$ for all $g' \in 2^{\mathrm{poly}(h)}$) are guaranteed to exist. But *what about exponentially long witnesses?* Are they guaranteed? Moreover, *what about sub-exponentially long witnesses, e.g., quasi-polynomially or polynomially long ones?*

It has long been known, through a theorem by Berman and Lingas [3], that a polynomially long witness would imply that $\mathsf{L} \not\supseteq \mathsf{NL}$. Here we strengthen that connection: a polynomially long witness would even imply that $\mathsf{L/poly} \not\supseteq \mathsf{NL}$; furthermore, the converse implication is also true.

Theorem 1. $\mathsf{2D} \not\supseteq \mathsf{2N}$ *has* poly(h)*-long witnesses iff* $\mathsf{L/poly} \not\supseteq \mathsf{NL}$.

Moreover, exponentially long witnesses and space $\log\log n$ are similarly linked.

Theorem 2. $\mathsf{2D} \not\supseteq \mathsf{2N}$ *has* $2^{\mathrm{poly}(h)}$*-long witnesses iff* $\mathsf{LL/polylog} \not\supseteq \mathsf{NLL}$.

Here, $\mathsf{LL/polylog} = \mathsf{NUDSPACE}(\log\log n)$ [8] is the class of languages recognizable in space $O(\log\log n)$ by DTMs with poly($\log n$) bits of advice, and NLL is the corresponding class for nondeterministic TMs (NTMs) and no advice.

These two theorems are the most representative ones in a list of variants that can be proved by similar arguments. One group of these variants focus on lengths of other growth rates. E.g., for quasi-polynomial lengths we get:

Theorem 3. *Let* $k \geq 1$ *and* $\varepsilon = 1/k$. *Then* $\mathsf{2D} \not\supseteq \mathsf{2N}$ *has* $2^{O(\log^k h)}$*-long witnesses iff* $\mathsf{DSPACE}(\log^\varepsilon n)/2^{O(\log^\varepsilon n)} \not\supseteq \mathsf{NSPACE}(\log^\varepsilon n)$.

Another group of variants focus on other modes of computation. E.g., consider *alternating* 2FAs and the corresponding complexity class 2A. As for 2N, we conjecture that $\mathsf{2D} \not\supseteq \mathsf{2A}$ and consider global upper bounds for the lengths of hard instances of an $\mathcal{L} \in \mathsf{2A} \setminus \mathsf{2D}$ over all families of poly(h)-state 2DFAs. We get:

Theorem 4. $\mathsf{2D} \not\supseteq \mathsf{2A}$ *has* poly(h)*-long witnesses iff* $\mathsf{L/poly} \not\supseteq \mathsf{P}$.

(Notice our use of the fact that $\mathsf{AL} = \mathsf{ASPACE}(\log n) = \mathsf{DTIME}\big(\mathrm{poly}(n)\big) = \mathsf{P}$.)

For the most part, the proofs of these theorems elaborate on standard, old ideas [3,12,18]. Perhaps their main value is what they imply for how we approach the two questions being connected. E.g., consider Theorems 1–3 and suppose that indeed $\mathsf{2D} \not\supseteq \mathsf{2N}$. On the one hand, people interested in size-bounded 2FAs can use our theorems to extract evidence about how hard it is for a certain proof strategy towards $\mathsf{2D} \not\supseteq \mathsf{2N}$ to succeed: e.g., a strategy that will eventually deliver a super-exponentially long witness is likely to succeed more easily than one that promises exponentially long witnesses, because the latter implies an additional breakthrough in understanding space-bounded NTMs, while the former does not. (All proof techniques currently available are of the former kind.) On the other hand, people interested in TMs of space $\Omega(\log\log n) \cap O(\log n)$ can find in the $\mathsf{2D} \vee \mathsf{2N}$ question a single unifying setting to work in: separating 2D and 2N with a super-exponentially long witness can be seen as the first step in a gradual approach that sees DTMs and NTMs separate for larger and larger space bounds as improved proof techniques establish shorter and shorter witnesses for $\mathsf{2D} \not\supseteq \mathsf{2N}$.

We conclude this introduction with a different strengthening of the Berman-Lingas Theorem, from [5]: in the case of unary automata $\mathsf{2D} \not\supseteq \mathsf{2N}$ implies $\mathsf{L} \not\supseteq \mathsf{NL}$ *irrespective of the lengths of hard instances*. Our research has been largely motivated by this recent theorem—and our title tries to reflect this.

2 Preparation

For $n \geq 1$, we let $[n] := \{0, \ldots, n-1\}$ and $\lg n := \max(1, \lceil \log_2 n \rceil)$. For S a set, $|S|$ denotes size. If Σ is an alphabet and $x \in \Sigma^*$, then $|x|$ is the length of x; x_i is its ith symbol; x^i is the concatenation of i copies of it; and $\langle x \rangle$ is its *binary encoding*, into $|x|$ blocks of $\lg |\Sigma|$ bits each (under a fixed ordering of Σ); ϵ is the empty string. A (*promise*) *problem* over Σ is a pair $L = (L^+, L^-)$ of disjoint subsets of Σ^*, the *positive* and *negative instances* of L; if $L^+ \cup L^- = \Sigma^*$ then L is a *language*. To *solve* L means to accept all $x \in L^+$ but no $x \in L^-$.

2.1 Machines

We assume familiarity with standard notation for 2FAs and TMs [16]. Our 2FAs consist of a finite *control* and a read-only *input tape*. Our TMs are 2FAs with two extra tapes: a read-only *advice tape* and a read-write *work tape*. Each tape is accessed via a dedicated *two-way head*. A *transducer* is a TM without advice tape but with a write-only *output tape* accessed via a *one-way head*. Details follow.

A (s, σ)-2NFA is any $A = (S, \Sigma, \delta, q_0, F)$ with $|S| = s$, $|\Sigma| = \sigma$, $q_0 \in S$, $F \subseteq S$, and $\delta \subseteq S \times (\Sigma \cup \{\vdash, \dashv\}) \times S \times \{\text{L},\text{R}\}$, where $\vdash, \dashv \notin \Sigma$ are endmarkers and L, R are directions. An $x \in \Sigma^*$ is presented on the input tape as $\vdash x \dashv$ and is considered accepted if δ allows a computation that starts at q_0 on $\vdash$ and eventually falls off $\dashv$ into a $q \in F$. The *language* of A is the set $L(A) := \{x \in \Sigma^* \mid A \text{ accepts } x\}$. The *binary encoding* of A is the string $\langle A \rangle := \mathtt{0}^s\mathtt{1}0^\sigma\mathtt{1}uvw$ where u, v, w encode δ, q_0, F with $2s^2(\sigma+2)$, $\lg s$, and s bits respectively (in the obvious ways, under fixed orderings of S and Σ). Note that $|\langle A \rangle| = O(s^2\sigma)$.

A (s, σ, γ)-NTM is any $M = (S, \Sigma, \Delta, \Gamma, \delta, q_0, q_f)$ with $|S|{=}s$, $|\Sigma|{=}\sigma$, $|\Gamma|{=}\gamma$, $q_0, q_f \in S$, and $\delta \subseteq S \times (\Sigma \cup \{\vdash, \dashv\}) \times (\Delta \cup \{\vdash, \dashv\}) \times (\Gamma \cup \{\sqcup\}) \times S \times \Gamma \times \{\text{L},\text{R}\}^3$, where $\sqcup \notin \Gamma$ is the blank symbol. An $x \in \Sigma^*$ and a $y \in \Delta^*$ are presented on the input and advice tapes as $\vdash x \dashv$ and $\vdash y \dashv$ respectively, and are considered accepted if δ allows a computation which starts at q_0, with blank work tape and the input and advice heads on $\vdash$, and eventually falls off $\dashv$ on the input tape into q_f. We say M is in *internal configuration* (q, i, j, w) if its state is q, its advice and work heads are at positions i and j, and its nonblank work tape content is $w \in \Gamma^*$.

For $\mathcal{Y} = (y_m)_{m \geq 0}$ a family of strings over Δ, the *language* of M under $\mathcal{Y}$ is $L(M, \mathcal{Y}) := \{x \in \Sigma^* \mid M \text{ accepts } x \text{ and } y_{|x|}\}$; if $\mathcal{Y}$ is the *empty advice* $(\epsilon)_{m \geq 0}$, we just write $L(M)$. We say $\mathcal{Y}$ is *strong advice* for M if, for all x and $m \geq |x|$, M accepts x and $y_{|x|}$ iff it accepts x and y_m. The *length* of $\mathcal{Y}$ is the function $m \mapsto |y_m|$. A function f is *strong space bound* for M under $\mathcal{Y}$ if, for all x and $m \geq |x|$, all computations on all x and y_m visit at most $f(m)$ work tape cells; if this is guaranteed just for the accepted x and y_m and then only for at least one accepting computation, then f is *weak*. Note that under *strong* advice y_m, M uses space $\leq f(m)$ and is correct on all x with $|x| \leq m$ (and not just $|x| = m$). This deviation from standard definitions is unimportant for advice of length $\text{poly}(m)$ or longer (because we can always replace y_m with the concatenation of $y_0, y_1, \ldots, y_m$). For shorter advice, though, it is not clear whether there is a difference. Our theorems need the strong version.

For $\mathcal{G}$ a set of functions, $\mathsf{NSPACE}(f)/\mathcal{G}$ consists of every $L(M, \mathcal{Y})$ where M is a NTM, $\mathcal{Y}$ is *strong* advice for M, with length in $\mathcal{G}$, and f is *strong* space bound for M under $\mathcal{Y}$. If $\mathcal{G} = \{0\}$ then only the empty advice is possible, and we just write $\mathsf{NSPACE}(f)$. Then $\mathsf{NL} := \mathsf{NSPACE}(\log n)$ and $\mathsf{NLL} := \mathsf{NSPACE}(\log\log n)$.

A machine is *deterministic* (2DFA, DTM) if its δ allows at most one computation per input. We define $\mathsf{DSPACE}(f)/\mathcal{G}$, $\mathsf{DSPACE}(f)$, L, LL analogously, and let $\mathsf{L/poly} := \mathsf{DSPACE}(\log n)/\mathrm{poly}(n)$, $\mathsf{LL/polylog} := \mathsf{DSPACE}(\log\log n)/\mathrm{poly}(\log n)$.

A function f is *fully space constructible* if there is a DTM which, on any input x and any advice, halts after visiting exactly $f(|x|)$ work tape cells.

Lemma 1. *Consider a* NTM M *under* strong *advice* $\mathcal{Y}$ *of length* g*, obeying a* weak *space bound* f*. For each length* m*, there is a* $2^{O(f(m)+\lg g(m))}$*-state* 2NFA A_m *that agrees with* M *under* $\mathcal{Y}$ *on all instances of length at most* m*.*

If M *is* deterministic*, then so is* A_m*. If* $\mathcal{Y}$ *is* empty*, then* A_m *also agrees with* M *under* $\mathcal{Y}$ *on all negative instances* (*of any length*)*. If* f *is* fully space constructible*, then there is a transducer* T *which, given* m (*in unary*) *and the corresponding* y_m*, computes* A_m (*in binary*) *in space* $O(f(m) + \lg g(m))$*.*

Proof. Let M be a (s, σ, γ)-NTM and $\mathcal{Y} = (y_m)_{m\geq 0}$, g, f as described. Fix m and consider M working on any input (of any length) and on y_m. Let S_m be the set of all internal configurations that can occur with $\leq f(m)$ work tape cells. Then

$$|S_m| \leq s(|y_m|+2)\textstyle\sum_{t=0}^{f(m)}(t+2)\gamma^t = O\big(g(m)f(m)^2\gamma^{f(m)}\big) = 2^{O(f(m)+\lg g(m))}\,.$$

We let $A_m := (S_m, \Sigma, \delta_m, q_0, F_m)$ where Σ is M's input alphabet, q_0 and F_m are the initial and accepting internal configurations in S_m, and $(c, a, c', d) \in \delta_m$ iff, reading $a \in \Sigma$, M under y_m can change its internal configuration from c to c' moving the input head towards d. Clearly, *if* M *is deterministic*, then so is A_m.

Now fix any input x (of any length). Let τ_m be the computation tree of M on x and y_m, and τ'_m the computation tree of A_m on x. It should be clear that each branch β in τ_m using $\leq f(m)$ work tape cells is fully simulated in τ'_m by an equally long branch β', which accepts iff β does. In contrast, each β using $> f(m)$ work tape cells is only partially simulated, by a shorter β' which hangs (at a rejecting state). Moreover, the β' of these two cases cover all branches in τ'_m.

Now suppose $|x| = n \leq m$. Let τ_n be the computation tree of M on x and y_n. We know τ_n accepts iff τ_m does (because $\mathcal{Y}$ *is strong*). Thus, if $x \in L(M, \mathcal{Y})$, then τ_n accepts and so does τ_m; hence, some accepting branch β in τ_m uses $\leq f(m)$ work tape cells (because f *is a weak space bound*); hence, its counterpart β' in τ'_m completes the simulation and accepts, too; so, A_m accepts x. Conversely, if $x \notin L(M, \mathcal{Y})$, then τ_n does not accept and neither does τ_m; hence, every branch in τ'_m is nonaccepting (because it either fully simulates a nonaccepting branch of τ_m or hangs after partially simulating one); so, A_m does not accept x.

If $\mathcal{Y}$ *is empty*, then A_m agrees also on all $x \notin L(M, \mathcal{Y})$ with $n > m$. Because then $y_n = y_m = \epsilon$ implies $\tau_n = \tau_m$, so τ_m does not accept, and neither can τ'_m.

If f *is fully space constructible*, then $\langle A_m\rangle$ can be computed from $\mathtt{0}^m$ and y_m, as follows. We first mark exactly $f(m)$ work tape cells, by running on $\mathtt{0}^m$ the DTM that fully constructs f. Using these cells and y_m, we then mark another

$\lg s + \lg(|y_m| + 2) + \lg\big(f(m) + 2\big)$ cells. Now the marked region is as long as the longest description of a $c \in S_m$. This makes it possible to iterate over all $c \in S_m$ or pairs thereof (by lexicographically iterating over all possible strings and discarding non-descriptions). From this point on, computing the bits of $\langle A_m \rangle$ is straightforward. Overall, $O(f(m) + \lg g(m))$ cells will suffice. □

Corollary 1. (a) *For each* NTM M *(under empty advice) of weak space* $O(\log n)$ *there is a log-space transducer which, given a length* m *(in unary), computes a* 2NFA A_m *(in binary) that has* $\mathrm{poly}(m)$ *states and may disagree with* M *only on positive instances longer than* m*. If* M *is deterministic, then so is* A_m*.*

(b) *For each* NTM M *(under empty advice) of weak space* $O(\log \log n)$ *and each length* m *there is a* $\mathrm{poly}(\log m)$*-state* 2NFA A_m *that may disagree with* M *only on positive instances longer than* m*. If* M *is deterministic, then so is* A_m*.*

2.2 Reductions

Let L_1, L_2 be problems over alphabets Σ_1, Σ_2. A *homomorphic reduction* of L_1 to L_2 [12] is any function $r : \Sigma_1 \cup \{\vdash, \dashv\} \to \Sigma_2^*$ such that $x \in L_1^+ \implies r(\vdash x \dashv) \in L_2^+$ and $x \in L_1^- \implies r(\vdash x \dashv) \in L_2^-$, where $r(\vdash x \dashv) := r(\vdash) r(x_1) \cdots r(x_{|x|}) r(\dashv)$. If there exists such an r, we write $L_1 \leq_{\mathrm{h}} L_2$. The *expansion* of r is the function $n \mapsto \max\{|r(\vdash x \dashv)| \mid x \in \Sigma_1^* \text{ and } |x| = n\}$. The *ternary encoding* $\langle r \rangle$ of r is the string $\langle r(\vdash)\rangle \# \langle r(a_1)\rangle \# \cdots \# \langle r(a_{\sigma_1})\rangle \# \langle r(\dashv)\rangle$, for $a_1, \ldots, a_{\sigma_1}$ a fixed ordering of Σ_1.

Lemma 2 ([12])**.** *If* $L_1 \leq_{\mathrm{h}} L_2$ *via a reduction* r *and* L_2 *is solvable by an* s*-state* 2NFA A_2*, then* L_1 *is solvable by a* $2s$*-state* 2NFA A_1*. If* A_2 *is* deterministic*, then so is* A_1*. Moreover, there exists a log-space transducer* T_{h} *which, given* r *(in ternary) and a deterministic* A_2 *(in binary), computes* A_1 *(in binary).*

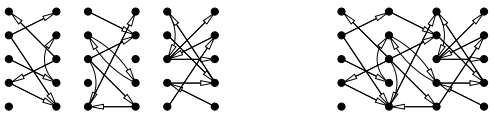
Proof. Let Σ_1 and Σ_2 be the alphabets of L_1 and L_2, and consider r and $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ as in the statement, where $|Q_2| = s$.

On input $x \in \Sigma_1^*$, the new automaton A_1 simulates A_2 on $r(\vdash x \dashv)$ piece-by-piece: on each symbol a on its tape, A_1 does what A_2 would eventually do on the corresponding infix $r(a)$ on its own tape (if a is $\vdash$ or $\dashv$, this corresponding infix is $\vdash r(\vdash)$ or $r(\dashv)\dashv$). This way, each branch β in the computation tree of A_2 on $r(\vdash x \dashv)$ is simulated by a branch in the computation tree of A_1 on x, which is accepting iff β is. Thus, x is accepted iff $r(\vdash x \dashv)$ is. Hence, if $x \in L_1^+$, then $r(\vdash x \dashv) \in L_2^+$ (by the selection of r), hence A_2 accepts, and so does A_1; if $x \in L_1^-$, then $r(\vdash x \dashv) \in L_2^-$ (as before), hence A_2 does not accept, and neither does A_1.

To perform this simulation, A_1 keeps track of the current state of A_2 and the side (L or R) from which A_2 enters the current corresponding infix. Formally, $A_1 := \big(Q_2 \times \{\mathrm{L},\mathrm{R}\}, \Sigma_1, \delta_1, (q_2,\mathrm{L}), F_2 \times \{\mathrm{L}\}\big)$, where δ_1 is easily derived from the above informal description. E.g., if $a \in \Sigma_1$ then $\big((p,\mathrm{L}), a, (q,\mathrm{L}), \mathrm{R}\big) \in \delta_1$ iff δ_2 allows on $r(a)$ a computation that starts at p on the leftmost symbol and eventually falls off the rightmost boundary into q (thus entering the following infix from its left side); and $\big((p,\mathrm{L}), \dashv, (q,\mathrm{R}), \mathrm{L}\big) \in \delta_1$ iff δ_2 allows on $r(\dashv)\dashv$ a computation that starts at p on the leftmost symbol and eventually falls off the leftmost boundary into q (thus entering the preceding infix from its right side).

If A_2 is deterministic, then clearly A_1 is also deterministic. Plus, $\langle A_1\rangle$ can be computed from $\langle r\rangle$ and $\langle A_2\rangle$ in logarithmic space. To see how, let $\sigma_1 := |\Sigma_1|$ and $\sigma_2 := |\Sigma_2|$, and recall that $\langle A_1\rangle = \mathtt{0}^{2s}\mathtt{1}0^{\sigma_1}\mathtt{1}u_1v_1w_1$ and $\langle A_2\rangle = \mathtt{0}^{s}\mathtt{1}0^{\sigma_2}\mathtt{1}u_2v_2w_2$, where the u, v, w encode respectively the transition functions, the start states, and the sets of final states. Each part of $\langle A_1\rangle$ can be computed from the corresponding part of $\langle A_2\rangle$ (using no work tape), except for $\mathtt{0}^{\sigma_1}$ and u_1. The former is computed from $\langle r\rangle$, by counting #s. For the latter, we mark $2\lg s+\lg(\sigma_1+2)+3 = O\big(\log(\sigma_1 s)\big)$ work tape cells and use them to iterate over all encodings of tuples $\big((p,d),a,(q,d'),d''\big)$ where $p,q \in [s]$ and $a \in \Sigma_1 \cup \{\vdash,\dashv\}$ and $d,d',d'' \in \{\text{L},\text{R}\}$, outputting 1 bit per tuple. To decide each bit, we locate $\langle r(a)\rangle$ in $\langle r\rangle$ and simulate A_2 on $r(a)$ (or on the appropriate endmarked version, if a is $\vdash$ or $\dashv$) starting at p on side d, until we either exceed $s\cdot|\langle r(a)\rangle|/\lg\sigma_2$ steps or fall off the string; if we fall off side d'' into q and $d' \neq d''$ then we output 1, otherwise we output 0. This simulation needs only a finite number of pointers into $\langle r\rangle$ and u_2, plus the aforementioned counter. Overall, the space used is logarithmic in $|\langle r\rangle+\langle A_2\rangle|$. □

For $h \geq 1$, the alphabet Γ_h consists of every directed graph with two columns of h nodes each (like the ones on the left, for $h = 5$). Easily, $|\Gamma_h| = 2^{(2h)^2}$. Each



$x \in \Gamma_h^*$ defines a multicolumn graph, derived by identifying adjacent columns (as shown on the right, for the three symbols on the left); if this graph contains a path from its leftmost to its rightmost column, we say x is *live*. We let $\text{TWL}_h := \{x \in \Gamma_h^* \mid x \text{ is live}\}$ (where 'TWL' abbreviates 'TWO-WAY LIVENESS'). Then the family $\text{TWL} := (\text{TWL}_h)_{h\geq 1}$ is 2N-complete [12], because of the following.

Lemma 3 ([12]). *Let A be any s-state* 2NFA. *Then $L(A) \leq_{\text{h}} \text{TWL}_{2s}$ via a reduction which has expansion $n+2$ and is log-space constructible from $\langle A\rangle$.*(3)

The *binary encoding* $\langle a\rangle$ of an $a \in \Gamma_h$ is a string of $(2h)^2$ bits that describes a (in the obvious way, under a fixed ordering of the $2h$ nodes). Using these encodings, we can "join" all languages TWL_h into the single binary language

$$\text{TWL JOIN} := \{\mathtt{0}^h\mathtt{1}\langle a_1\rangle\langle a_2\rangle\cdots\langle a_l\rangle\mathtt{1}0^t \mid h \geq 1 \ \&\ h \text{ divides } t \ \&\ l \geq 0 \ \&\ \text{each } a_i \in \Gamma_h \ \&\ a_1a_2\cdots a_l \text{ is live}\},$$

where the h leading 0s determine the interpretation: the leftmost and rightmost 1s must be separated by 0 or more $(2h)^2$-long blocks of bits (the "middle bits"), and the multicolumn graph described by these blocks must contain a path from its leftmost to its rightmost column; finally, the number t of trailing 0s must be a multiple of h.(4) If this last condition that "h divides t" is replaced by the stronger condition that "every $j \leq h$ divides t" ("Szepietowski-padding" [18]), we get an alternative join, which we call TWL LONG-JOIN.

Let $\text{A}_{\text{2NFA}} := \{\langle A\rangle\#\langle x\rangle \mid A \text{ is a 2NFA and } A \text{ accepts } x\}$ be the *acceptance problem* for 2NFAs, and A_{2DFA} the corresponding problem for 2DFAs. Note that no alphabet is fixed: solving these problems involves checking that the bits after # can be interpreted as an input for the 2FA that is encoded by the bits before #.

Lemma 4. *Let A be an (s,σ)-2NFA. Then $L(A) \leq_{\text{h}}$-reduces to each of A_{2NFA}, TWL JOIN, and TWL LONG-JOIN via reductions of expansion $O(s^2\sigma n)$, $O(s^2 n)$, and $2^{O(s)}n$.(5) The first two reductions are log-space constructible from $\langle A\rangle$.*

Proof. For $\Sigma = \{a_0, \ldots, a_{\sigma-1}\}$ the alphabet of A, let r_1 be the homomorphism that maps $\vdash$ to $\langle A\rangle\#$, each a_i to the $\lg\sigma$-long binary code of i, and $\dashv$ to ϵ. Then each n-long x is sent to $r_1(\vdash x\dashv) = \langle A\rangle\#\langle x\rangle$, of length $(|\langle A\rangle|+1) + n\lg\sigma + 0 = O(s^2\sigma) + n\lg\sigma = O(s^2\sigma n)$, which (clearly) is in A_{2NFA} iff $x \in L(A)$.

For $r : \Sigma \cup \{\vdash,\dashv\} \to \Gamma_{2s}$ the reduction from Lemma 3, let r_2 be the homomorphism that maps $\vdash$ to $\mathtt{0}^{2s}\mathtt{1}\langle r(\vdash)\rangle$, each a_i to $\langle r(a_i)\rangle$, and $\dashv$ to $\langle r(\dashv)\rangle\mathtt{10}^{2s}$. Then each n-long x is sent to $r_2(\vdash x\dashv) = \mathtt{0}^{2s}\mathtt{1}\langle r(\vdash)\rangle\langle r(x_1)\rangle\cdots\langle r(x_n)\rangle\langle r(\dashv)\rangle\mathtt{10}^{2s}$, of length $(2s{+}1) + (n{+}2)(4s)^2 + (1{+}2s) = O(s^2 n)$, which is in TWL JOIN iff $r(\vdash)r(x_1)\cdots r(x_n)r(\dashv)$ is live (all other conditions in the definition of TWL JOIN are satisfied), namely iff $r(\vdash x\dashv) \in \text{TWL}_{2s}$, which is true iff $x \in L(A)$.

Let r_3 be the homomorphism that differs from r_2 only in that it maps $\dashv$ to $\langle r(\dashv)\rangle\mathtt{10}^{\lambda(2s)}$, where $\lambda(2s)$ is the least common multiple of all $j = 1, \ldots, 2s$. Since $\lambda(2s) = 2^{\Theta(s)}$ [18, Lemma, part (b)], now each n-long x is sent to $r_3(\vdash x\dashv)$, of length $(2s{+}1) + (n{+}2)(4s)^2 + (1{+}2^{\Theta(s)}) = 2^{O(s)}n$, which (as before) is in TWL LONG-JOIN iff $r(\vdash)r(x_1)\cdots r(x_n)r(\dashv)$ is live, and thus iff $x \in L(A)$. □

Lemma 5. A_{2NFA} *and* TWL JOIN *are* NL*-complete and* TWL LONG-JOIN $\in$ NLL.

Proof. That A_{2NFA} is NL-complete is well-known; in fact, NL-hardness follows from Corollary 1a and Lemma 4. The NL-hardness of TWL JOIN follows similarly.

To solve TWL JOIN by a NTM in space $O(\log n)$, we work in two stages. First, we deterministically verify the format: we check that there are at least two 1s, count the number h of leading 0s, check that h divides the number of trailing 0s (by scanning and counting modulo h) and that $(2h)^2$ divides the number of "middle bits" (similarly). Then, we nondeterministically verify liveness, by simulating on the "middle bits" the $2h$-state 2NFA solving TWL_h [12]. Each stage can be performed in space $O(\log h)$. Since $h \leq n$, TWL JOIN $\in$ NL.

To solve TWL LONG-JOIN we use the same algorithm, but with a preliminary stage [18]. This starts by incrementing a counter from 1 upto the first number, $\tilde{t}$, that does not divide the number t of trailing 0s (divisibility is always tested by scan-and-count, as above). On reaching $\tilde{t}$, we compare it with the number h of leading 0s (scan-and-count, again). If $\tilde{t} \leq h$, we reject; otherwise we continue to the two main stages. Correctness should be clear. The space used in the preliminary stage is $O(\log\tilde{t})$. The two main stages cost $O(\log h)$, as above, which is also $O(\log\tilde{t})$ because we get to them only if $h < \tilde{t}$. Since $\tilde{t} = O(\log t)$ [18, Lemma, part (d)] and $t \leq n$, the total space used is $O(\log\log n)$. □

3 The Berman-Lingas Theorem

The Berman-Lingas Theorem [3, Th. 6] is usually cited as: *if* $\mathsf{L} = \mathsf{NL}$ *then every* s*-state* 2NFA *can be simulated on* $\mathrm{poly}(s)$*-long inputs by some* $\mathrm{poly}(s)$*-state* 2DFA. However, Berman and Lingas actually claim a stronger statement [3, p. 17]:[(6)]

> $\mathsf{L} = \mathsf{NL}$ *iff for each alphabet* Σ *there exists a log-space* DTM T_Σ *which, on input a* 2NFA A *over* Σ *(in binary) and a length* m *(in unary), outputs a* 2DFA B *(in binary) that has* $\mathrm{poly}(sm)$ *states, for* s *the number of states in* A*, and may disagree with* A *only on positive instances longer than* m*.*

Namely, the promised 2DFA is log-space constructible from the 2NFA and the length bound and, with this into account, the converse is also true. Then, the preceding citation is a corollary for the special case where $m = \mathrm{poly}(s)$.

However, these statements are valid only if Σ is constant. If instead the alphabet grows with s, which is true for some $\mathsf{2N}$-complete problems, then the bound for the states in B might even be exponential in s. To highlight this subtle point, we state and prove the theorem under no assumptions for alphabet size.

Theorem (Berman-Lingas [3]). $\mathsf{L} \supseteq \mathsf{NL}$ *iff there exists a log-space* DTM T *which, on input a* 2NFA A *(in binary) and a length* m *(in unary), outputs a* 2DFA B *(in binary) that has* $\mathrm{poly}(s\sigma m)$ *states, for* s *and* σ *the number of states and symbols in* A*, and may disagree with* A *only on positive instances longer than* m*.*

Proof. [⇒] Suppose $\mathsf{L} \supseteq \mathsf{NL}$. Then $\mathrm{A}_{\text{2NFA}} \in \mathsf{L}$. Hence, some DTM M solves A_{2NFA} under empty advice and in space strongly bounded by $\log n$.

Pick any (s, σ)-2NFA A and length m. By Lemma 4, we know $L(A) \leq_{\mathrm{h}} \mathrm{A}_{\text{2NFA}}$ via a reduction that has expansion $\mu(n) = O(s^2 \sigma n)$ and is constructible from $\langle A \rangle$. Hence (Lemma 2), constructing a 2DFA B for $L(A)$ and lengths $\leq m$ reduces to constructing a 2DFA $\tilde{B}$ for $L(M)$ and lengths $\leq \mu(m)$. This latter construction is indeed possible, by Corollary 1a. We thus employ the series of transductions:

$$\langle A \rangle \#\mathsf{0}^m \xrightarrow{T_1} \langle A \rangle \#\mathsf{0}^{\mu(m)} \xrightarrow{T_M} \langle A \rangle \# \langle \tilde{B} \rangle \xrightarrow{T_2} \langle r \rangle \# \langle \tilde{B} \rangle \xrightarrow{T_{\mathrm{h}}} \langle B \rangle \,.$$

First, a simple log-space transducer T_1 converts the original input $\langle A \rangle \#\mathsf{0}^m$ into $\langle A \rangle \#\mathsf{0}^{\mu(m)}$, by doing the algebra. Then the log-space transducer T_M guaranteed by Corollary 1a for M converts $\mathsf{0}^{\mu(m)}$ into a 2DFA $\tilde{B}$ that has $\mathrm{poly}\big(\mu(m)\big)$ states and can disagree with M only on positive instances longer than $\mu(m)$. Then the log-space transducer T_2 guaranteed by Lemma 4 for $L(A) \leq_{\mathrm{h}} \mathrm{A}_{\text{2NFA}}$ converts $\langle A \rangle$ into the encoding of the underlying homomorphism r. Finally, the log-space transducer T_{h} from Lemma 2 converts $\langle r \rangle \# \langle \tilde{B} \rangle$ into the desired 2DFA B of $2 \cdot \mathrm{poly}\big(\mu(m)\big) = \mathrm{poly}(s\sigma m)$ states. By the transitivity of log-space transductions, the full algorithm can also be implemented in logarithmic space.

[⇐] Suppose the log-space DTM T of the statement exists. Then A_{2NFA} can be reduced to A_{2DFA} in logarithmic space, which implies $\mathsf{L} \supseteq \mathsf{NL}$ (since A_{2NFA} is NL-hard, $\mathrm{A}_{\text{2DFA}} \in \mathsf{L}$, and L is closed under log-space reductions). The reduction works in two log-space steps. We first run a simple log-space transducer to convert the

input $\langle A\rangle\#\langle x\rangle$ into $\langle A\rangle\#0^m\#\langle x\rangle$, where $m := |x| = |\langle x\rangle|/\lg\sigma$, for σ the size of the alphabet of A. We then apply T on $\langle A\rangle\#0^m$ to produce $\langle B\rangle\#\langle x\rangle$, where B a 2DFA that agrees with A on all inputs of length at most m, including x. □

It is now clear that, in the special case where we restrict to inputs of length $m = \text{poly}(s)$, the bound for the states in B is just $\text{poly}(s\sigma)$. Hence, for an A over an alphabet of size $\sigma = 2^{\Omega(s)}$ (e.g., a 2NFA for TWL_s), our bound for the states in a 2DFA simulating A on $\text{poly}(s)$-long inputs is just $2^{\text{poly}(s)}$. We stress that this looseness is inherent in the Berman-Lingas argument, and not just in our analysis of it: a larger alphabet for A implies longer encoding $\langle A\rangle$, thus longer inputs to the alleged log-space DTM M for A_{2NFA}, more space available for M on its work tape, more internal configurations, more states in the 2DFA $\tilde{B}$ simulating M, and thus more states in the final 2DFA B simulating $\tilde{B}$.

It is also clear that the above theorem cannot be an equivalence without a constructive relationship between the given 2NFA A and length m and the promised 2DFA B. That is, if in the converse direction we are promised that for each A and m a fitting B simply *exists*, as opposed to it being log-space constructible, then the argument fails. This is an additional obstacle in connecting with the 2D v 2N question, which is purely existential: a proof that $\mathsf{2D} \supseteq \mathsf{2N}$ needs no log-space conversion of s-state 2NFAs to equivalent $\text{poly}(s)$-state 2DFAs.

The next theorem removes both of the above dependences, on alphabet size and on log-space constructibility. To remove dependence on alphabet size, we switch to another NL-complete problem; to remove dependence on constructibility, we switch to nonuniform L. The main structure of the argument is similar.

Theorem 1. $\mathsf{L/poly} \supseteq \mathsf{NL}$ *iff for every* 2NFA A *and length* m *there is a* 2DFA B *that has* $\text{poly}(sm)$ *states, for* s *the number of states in* A*, and agrees with* A *on all instances of length at most* m*.*

Proof. [⇒] Suppose $\mathsf{L/poly} \supseteq \mathsf{NL}$. Then TWL JOIN $\in \mathsf{L/poly}$. Hence, some DTM M solves TWL JOIN under strong advice $\mathcal{Y}$ of length $g(m) = \text{poly}(m)$ in space strongly bounded by $f(m) = \log m$.

Pick any (s,σ)-2NFA A and length m. By Lemma 4, $L(A) \leq_{\text{h}} \text{TWL JOIN}$ with expansion $\mu(n) = O(s^2 n)$. Hence (Lemma 2), a $\text{poly}(sm)$-state 2DFA B for $L(A)$ and lengths $\leq m$ exists if there exists a $\text{poly}(sm)$-state 2DFA $\tilde{B}$ for $L(M,\mathcal{Y})$ and lengths $\leq \mu(m)$. Such a $\tilde{B}$ is given by Lemma 1 for M and $\mathcal{Y}$: both $f(\mu(m))$ and $\lg g\big(\mu(m)\big)$ are $O\big(\log(sm)\big)$, and thus $2^{O(f(\mu(m))+\lg g(\mu(m)))} = \text{poly}(sm)$.

[⇐] Suppose that for every s-state 2NFA A and length m there is a $\text{poly}(sm)$-state 2DFA B that agrees with A on all instances of length $\leq m$.

Pick any $L \in \mathsf{NL}$. Let Σ be its alphabet and M some NTM that solves it (under empty advice) in space strongly bounded by $\log n$. We know (Corollary 1a) that for each length m there is a $\text{poly}(m)$-state 2NFA A_m over Σ that agrees with M on lengths $\leq m$. Applying our assumption to each A_m and m, we find a 2DFA B_m over Σ that has $\text{poly}(\text{poly}(m)\cdot m) = \text{poly}(m)$ states and agrees with A_m on lengths $\leq m$, meaning that it decides L correctly on lengths $\leq m$.

Now let $\mathcal{Y} = (y_m)_{m\geq 0} := (\langle B_m\rangle)_{m\geq 0}$ and consider the DTM U over Σ which, on input x and advice y, reads y as the encoding of a 2DFA and simulates it on x.

Then $L(U, y) = \{x \in \Sigma^* \mid U \text{ accepts } x \text{ and } y_{|x|}\} = \{x \in \Sigma^* \mid B_{|x|} \text{ accepts } x\} = \{x \in \Sigma^* \mid x \in L\} = L$. Note that y is *strong* advice: if $|x| \leq m$ then U accepts x and $y_m \Leftrightarrow B_m$ accepts $x \Leftrightarrow x \in L \Leftrightarrow B_{|x|}$ accepts $x \Leftrightarrow U$ accepts x and $y_{|x|}$. Also, $|y_m| = |\langle B_m \rangle| = O(\text{poly}(m)^2 \cdot |\Sigma|) = \text{poly}(m)$ since Σ is constant, and the simulation uses space strongly logarithmic in the number of states in B_m, namely $O\big(\log \text{poly}(m)\big) = O(\log m)$, irrespective of x. Overall, $L \in \mathsf{L/poly}$. □

We remark that the two dependences, on alphabet size and constructibility, do not need to be removed from the Berman-Lingas Theorem simultaneously. Each can also be removed individually, leading to two more variants of the original theorem: one that differs from it only in that the bound for B is $\text{poly}(sm)$, and one that differs from Theorem 1 only in that the bound for B is $\text{poly}(s\sigma m)$.

Our second extension of the Berman-Lingas Theorem reduces the dependence on the input lengths exponentially, by connecting to the stronger statement that nondeterminism can be nonuniformly removed even on the $\log \log n$ level. The argument is again similar, we just need to switch to the long join of TWL.

Theorem 2. $\mathsf{LL/polylog} \supseteq \mathsf{NLL}$ *iff for every* 2NFA A *and length* m *there is a* 2DFA B *that has* $\text{poly}(s \log m)$ *states, for* s *the number of states in* A*, and agrees with* A *on all instances of length at most* m*.*

Proof. [⇒] If $\mathsf{LL/polylog} \supseteq \mathsf{NLL}$ then TWL LONG-JOIN $\in \mathsf{LL/polylog}$, so a DTM M solves TWL LONG-JOIN under strong advice y of length $g(m) = \text{poly}(\log m)$ in space strongly bounded by $f(m) = \log \log m$. As above, for any (s, σ)-2NFA A and length m, we know $L(A) \leq_{\text{h}}$ TWL LONG-JOIN via a reduction of expansion $\mu(n) = 2^{O(s)}n$ (Lemma 4). So (Lemma 2), it suffices to find a $\text{poly}(s \log m)$-state 2DFA $\tilde{B}$ for $L(M, y)$ on lengths $\leq \mu(m)$. This is given by Lemma 1, as both $f(\mu(m))$ and $\lg g\big(\mu(m)\big)$ are $O\big(\log(s + \log m)\big)$, and so $2^{O(f(\mu(m)) + \lg g(\mu(m)))} = \text{poly}(s \log m)$.

[⇐] Suppose for each s-state 2NFA A and length m there is a $\text{poly}(s \log m)$-state 2DFA B that agrees with A on lengths $\leq m$. As above, we pick any $L \in \mathsf{NLL}$ over some Σ and let M be a NTM solving L in space strongly bounded by $\log \log n$. Corollary 1b guarantees that for each m some $\text{poly}(\log m)$-state 2NFA A_m over Σ agrees with M on lengths $\leq m$. Hence (by our assumption), there is also a 2DFA B_m over Σ with $\text{poly}(\text{poly}(\log m) \cdot \log m) = \text{poly}(\log m)$ states that agrees with A_m on lengths $\leq m$, and thus correctly decides L on these lengths. So, we consider $y = (y_m)_{m \geq 0} := (\langle B_m \rangle)_{m \geq 0}$. Under this advice, the same DTM U as above (over the specific Σ) solves L. The advice is *strong* for the same reasons, but its length now is $|y_m| = |\langle B_m \rangle| = O(\text{poly}(\log m)^2 \cdot |\Sigma|) = \text{poly}(\log m)$, causing the space usage to be $O\big(\log \text{poly}(\log m)\big) = O(\log \log m)$. □

A third extension is proved via the variants of TWL JOIN where the padding condition "h divides t" is replaced by "every $j \leq \log^k h$ divides t", for $k \geq 1$.

Theorem 3. $\mathsf{DSPACE}(\log^\varepsilon n)/2^{O(\log^\varepsilon n)} \supseteq \mathsf{NSPACE}(\log^\varepsilon n)$*, where* $\varepsilon = 1/k \leq 1$*, iff for every* 2NFA A *and length* m *there is a* 2DFA B *that has* $\text{poly}(s2^{\log^\varepsilon m})$ *states, for* s *the number of states in* A*, and agrees with* A *on all instances of length at most* m*.*

4 Conclusion

We can now observe that Theorems 1-3 of the previous section are respectively equivalent to Theorems 1-3 of the Introduction. For the pair of Theorems 1 this is shown by the following lemma. Similar lemmas are possible for the other pairs.

Lemma 6. $\mathsf{2D} \not\supseteq \mathsf{2N}$ *has no* $\mathrm{poly}(h)$*-long witnesses iff for every* 2NFA A *and length* m *there is a* 2DFA B *that has* $\mathrm{poly}(sm)$ *states, for* s *the number of states in* A*, and agrees with* A *on all instances of length at most* m*.*

Proof. [⇒] Suppose $\mathsf{2D} \not\supseteq \mathsf{2N}$ has no $\mathrm{poly}(h)$-long witnesses. In particular, TWL is not an h-long witness. Hence, some family $\mathcal{B}' = (B'_h)_{h\geq 1}$ of $\mathrm{poly}(h)$-state 2DFAs admits no h-long hard instances of TWL: i.e., for each family $x = (x_h)_{h\geq 1}$ of such instances, $\mathcal{B}'$ errs on x only finitely often. Hence, only finitely many B'_h fail to be correct on all $\leq h$-long instances of TWL_h. Replacing the failing B'_h with larger 2DFAs that do not fail, we get a new family $\mathcal{B} = (B_h)_{h\geq 1}$ of $\mathrm{poly}(h)$-state 2DFAs where B_h decides TWL_h correctly on all $\leq h$-long instances, for all h.

Pick any s-state 2NFA A and length m. Let $\tau := \max(2s, m{+}2)$. By Lemma 3, we know $L(A) \leq_{\mathrm{h}} \mathrm{TWL}_{2s}$ with expansion $\mu(n) = n{+}2$. Also, $\mathrm{TWL}_{2s} \leq_{\mathrm{h}} \mathrm{TWL}_\tau$ with expansion $\mu'(n) = n$ (simply map $\vdash, \dashv$ to ϵ, and each $a \in \Gamma_{2s}$ to the $a' \in \Gamma_\tau$ that has only the arrows of a). So, $L(A) \leq_{\mathrm{h}} \mathrm{TWL}_\tau$ with expansion $\mu'(\mu(n))$. So (Lemma 2), a 2DFA B for $L(A)$ on lengths $\leq m$ can be derived from a 2DFA for TWL_τ on lengths $\leq \mu'(\mu(m)) = m{+}2 \leq \tau$. Such a 2DFA is B_τ, with $\mathrm{poly}(\tau)$ states. Hence, B has $2 \cdot \mathrm{poly}(\tau) \leq 2 \cdot \mathrm{poly}(2s + (m{+}2)) = \mathrm{poly}(sm)$ states.

[⇐] Suppose for each s-state 2NFA and length m there is a $\mathrm{poly}(sm)$-state 2DFA agreeing on all $\leq m$-long instances. Pick any $\mathcal{L} = (L_h)_{h\geq 1} \in \mathsf{2N}$ and $p \in \mathrm{poly}(h)$. Since $\mathcal{L} \in \mathsf{2N}$, there is a $q \in \mathrm{poly}(h)$ and a family $\mathcal{A} = (A_h)_{h\geq 1}$ of $q(h)$-state 2NFAs solving $\mathcal{L}$. Applying our assumption on each A_h and $p(h)$, we find a 2DFA B_h that agrees with A_h on all $\leq p(h)$-long instances and has $\mathrm{poly}(q(h)p(h)) = \mathrm{poly}(h)$ states. Hence, $(B_h)_{h\geq 1}$ is a family of $\mathrm{poly}(h)$-state 2DFAs that admit no family of $p(h)$-long hard instances of $\mathcal{L}$. □

In closing, we suggest some notation that may facilitate discussions like ours. Consider any 2FA family $\mathcal{A} = (A_h)_{h\geq 1}$, any family of promise problems $\mathcal{L} = (L_h)_{h\geq 1}$, and any two function classes $\mathcal{F}, \mathcal{G}$. We say $\mathcal{A}$ is $\mathcal{F}$*-large* if A_h has $\leq f(h)$ states, for some $f \in \mathcal{F}$ and all h; and that $\mathcal{L}$ is $\mathcal{G}$*-long* if $|x| \leq g(h)$, for some $g \in \mathcal{G}$ and all h and $x \in L_h^+ \cup L_h^-$. Then the class $\mathsf{2NSIZE}(\mathcal{F})/\mathcal{G}$ consists of every $\mathcal{G}$-long family of promise problems solvable by some $\mathcal{F}$-large family of 2NFAs —if we just write $\mathsf{2N}/\mathcal{G}$, we mean $\mathcal{F} = \mathrm{poly}(h)$; if we just write $\mathsf{2NSIZE}(\mathcal{F})$, we mean $\mathcal{G}$ contains all functions. We specifically let $\mathsf{2N/poly} := \mathsf{2N}/\mathrm{poly}(h)$ and $\mathsf{2N/exp} := \mathsf{2N}/2^{\mathrm{poly}(h)}$. For 2DFAs, we define the classes $\mathsf{2DSIZE}(\mathcal{F})/\mathcal{G}$, $\mathsf{2D}/\mathcal{G}$, $\mathsf{2DSIZE}(\mathcal{F})$ analogously. Now, Theorems 1, 3, and 2 can be restated as:

$$\begin{array}{lcl} \mathsf{2D} \supseteq \mathsf{2N/poly} & \iff & \mathsf{L/poly} \supseteq \mathsf{NL} \\ \mathsf{2D} \supseteq \mathsf{2N}/2^{O(\log^k n)} & \iff & \mathsf{DSPACE}(\log^\varepsilon n)/2^{O(\log^\varepsilon n)} \supseteq \mathsf{NSPACE}(\log^\varepsilon n) \\ \mathsf{2D} \supseteq \mathsf{2N/exp} & \iff & \mathsf{LL/polylog} \supseteq \mathsf{NLL} \end{array}$$

where $\varepsilon = 1/k \leq 1$. Note that, next to a TM complexity class, "$/\cdot$" bounds the advice length; next to a 2FA complexity class, it bounds the length of the inputs.

References

1. M. Alberts. Space complexity of alternating Turing machines. In *Proceedings of FCT*, pages 1–7, 1985.
2. R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of FOCS*, pages 218–223, 1979.
3. P. Berman and A. Lingas. On complexity of regular languages in terms of finite automata. Report 304, Institute of Computer Science, Polish Academy of Sciences, Warsaw, 1977.
4. V. Geffert, C. Mereghetti, and G. Pighizzini. Converting two-way nondeterministic unary automata into simpler automata. *Theoretical Computer Science*, 295:189–203, 2003.
5. V. Geffert and G. Pighizzini. Two-way unary automata versus logarithmic space. In *Proceedings of DLT*, pages 197–208, 2010.
6. J. E. Hopcroft and J. D. Ullman. Some results on tape-bounded Turing machines. *Journal of the ACM*, 16(1):168–177, 1967.
7. J. Hromkovič and G. Schnitger. Nondeterminism versus determinism for two-way finite automata: generalizations of Sipser's separation. In *Proceedings of ICALP*, pages 439–451, 2003.
8. O. H. Ibarra and B. Ravikumar. Sublogarithmic-space Turing machines, nonuniform space complexity, and closure properties. *Mathematical Systems Theory*, 21:1–17, 1988.
9. C. Kapoutsis. Deterministic moles cannot solve liveness. *Journal of Automata, Languages and Combinatorics*, 12(1-2):215–235, 2007.
10. R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of STOC*, pages 302–309, 1980.
11. S. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7:207–223, 1964.
12. W. J. Sakoda and M. Sipser. Nondeterminism and the size of two-way finite automata. In *Proceedings of STOC*, pages 275–286, 1978.
13. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.
14. J. I. Seiferas. Untitled manuscript, communicated to M. Sipser. Oct. 1973.
15. M. Sipser. Lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 21(2):195–202, 1980.
16. M. Sipser. *Introduction to the theory of computation.* PWS Publishing Company, Boston, MA, 1996.
17. R. E. Stearns, J. Hartmanis, and P. M. Lewis II. Hierarchies of memory limited computations. In *Proceedings of SWCT*, pages 179–190, 1965.
18. A. Szepietowski. If deterministic and nondeterministic space complexities are equal for $\log\log n$ then they are also equal for $\log n$. In *Proceedings of STACS*, pages 251–255, 1989.

Notes

[(1)]Note the unusual form '$\mathsf{A} \not\supseteq \mathsf{B}$'. Whenever $\mathsf{A} \subseteq \mathsf{B}$, this is of course equivalent to the more familiar '$\mathsf{A} \subsetneq \mathsf{B}$'. Still, we will be studying cases where $\mathsf{A} \subseteq \mathsf{B}$ is false (e.g., $\mathsf{A} = \mathsf{L/poly}$ & $\mathsf{B} = \mathsf{NL}$) and thus '$\mathsf{A} \not\supseteq \mathsf{B}$' is appropriate. We thus stick to

it throughout the article, so that every statement is easy to compare with any other. We read and think of '$\mathsf{A} \not\supseteq \mathsf{B}$' as '$\mathsf{A}$ does not cover B' or 'B eludes A'.

(2) *Proof* (Lemma 0). The *behavior* of a 2NFA A on a string x is the set of tuples (p, d, q, e) of states p, q and sides $d, e \in \{\text{L},\text{R}\}$ such that A can exhibit a computation that starts at p on the $d$most symbol of x and eventually falls off the $e$most symbol of x into q. Easily, if A has s states, then it can exhibit at most $2^{(2s)^2}$ distinct behaviors. For a 2DFA, this number is $(2s+1)^{2s} = 2^{2s \lg(2s+1)}$.

Now suppose $\mathcal{L} \in \mathsf{2N} \setminus \mathsf{2D}$. Pick any 2DFA family $\mathcal{B} = (B_h)_{h \geq 1}$ where B_h has $O(h^b)$ states, for a constant b. Since $\mathcal{L} \notin \mathsf{2D}$, $\mathcal{B}$ admits a family $\mathcal{x} = (x_h)_{h \geq 1}$ of hard instances: B_h errs on x_h for infinitely many h. Since $\mathcal{L} \in \mathsf{2N}$, there is a 2NFA family $\mathcal{A} = (A_h)_{h \geq 1}$ where A_h solves L_h with $O(h^a)$ states, for a constant a.

The number of distinct behaviors that A_h and B_h can exhibit on a particular input are respectively $2^{O(h^{2a})}$ and $2^{O(h^b \log h)}$. Hence, the number g_h of distinct pairs of behaviors of A_h and B_h is $2^{O(h^{2a} + h^b \log h)} = 2^{O(h^c)}$, if $c = \max(2a, b+1)$. If x_h is longer than g_h, then it contains two prefixes on which A_h behaves the same and B_h behaves the same, too. Removing the infix of x_h between the right boundaries of these prefixes, we get a shorter input x'_h that neither A_h nor B_h can distinguish from x_h. In particular, B_h errs on x'_h iff it errs on x_h. Repeating this process, we eventually bring the length of x'_h below g_h.

The family of inputs $\mathcal{x}' = (x'_h)_{h \geq 1}$ obtained this way is also a family of hard instances of $\mathcal{L}$ for $\mathcal{B}$, and their lengths are $2^{O(h^c)} = 2^{\text{poly}(h)}$. Notice that, for a family $\mathcal{B}'$ of polynomially larger 2DFAs (i.e., $b' > b$ and $b' \geq 2a$), our argument will return a family of super-polynomially longer hard instances (i.e., $c' > c$ —to be precise, they will be quasi-polynomially longer). □

(3) *Proof* (Lemma 3). The reduction maps each $a \in \Sigma \cup \{\vdash, \dashv\}$ to a single symbol $r(a) \in \Gamma_{2s}$ that fully encodes the "behavior of A on a" (as defined in Note 2, see [12] for details). Hence, $|r(\vdash x \dashv)| = |x| + 2$ for all $x \in \Sigma^*$. Constructing $\langle r \rangle$ out of $\langle A \rangle$ is easily done with a finite number of pointers into $\langle A \rangle$. □

(4) The suffix $\mathtt{1}\mathtt{0}^t$ and the condition 'h divides t' are redundant. They are included only for symmetry, as the respective suffix and condition are needed in the variants of TWL JOIN (TWL LONG-JOIN and the one described before Theorem 3).

(5) More tightly, the first expansion is $O(s^2\sigma) + n \log \sigma$. But the looser $O(s^2 \sigma n)$ is simpler and does not harm conclusions, since it is later ([⇒] of B-L Theorem) fed to an unspecified polynomial. Similar looseness is adopted elsewhere, too.

(6) The actual statement of [3, Th. 6] is very close to the usual citation given in our text. However, it also includes a pointer to a Remark on p. 17, which explains that the promised 2DFA can be constructed in logarithmic space and that with this observation the theorem becomes an equivalence.

Also note that the second, complete statement in our text uses m as the length bound. The actual statement of [3, Th. 6] uses $s \cdot m$, for s the number of states in the 2NFA. These two statements are equivalent. We have opted for the one that is simpler and facilitates comparison with subsequent statements.