A Characterisation of NL/*poly* via Nondeterministic Finite Automata.

Rob Myers and Henning Urbat

Institut für Theoretische Informatik
TU Braunschweig, Germany

Abstract. For each language $L \subseteq \mathbf{2}^*$ and function $t : \mathbb{N} \to \mathbb{N}$, we define another language $t * L \subseteq \mathbf{2}^*$. We then prove that $L \in \mathsf{NL}/poly$ if and only if there exists $k \in \mathbb{N}$ such that the projections $(n^k * L) \cap \mathbf{2}^n$ are accepted by nondeterministic finite automata of size polynomial in n. Therefore, proving super-polynomial lower bounds for *unrestricted nondeterministic branching programs* reduces to proving super-polynomial lower bounds for *oblivious read-once nondeterministic branching programs* i.e. nondeterministic finite automata.

1 Introduction

In this paper, we show that proving super-polynomial lower bounds for nondeterministic branching programs (nbps) is essentially equivalent to proving them for nondeterministic finite automata (nfas). It is a major open problem in complexity theory to provide an explicit language $L \subseteq \mathbf{2}^*$, such that there is no sequence of nbps $\mathcal{B}_n$ of size polynomial in n accepting the projections $L_n = L \cap \mathbf{2}^n$ [4]. However, if one restricts to sequences of nfas – which are instances of *oblivious read-once* nbps – many explicit languages with provably super-polynomial lower bounds are known, e.g. the language of all binary palindromes. Since the latter has linear size nbps, which read the variables in the appropriate order, we cannot simply replace nbps by nfas. Instead, given any $L \subseteq \mathbf{2}^*$ and function $t : \mathbb{N} \to \mathbb{N}$ we define another language $t * L \subseteq \mathbf{2}^*$. We then show that for any language L, the projections L_n are accepted by nbps of size polynomial in n iff there exists $k \in \mathbb{N}$ such that the projections $(n^k * L)_n$ are accepted by nfas of size polynomial in n. This is achieved via a relatively simple translation between nbps and nfas.

Recall the non-uniform complexity class $\mathsf{NL}/poly$ i.e. those languages accepted by a single logspace-bounded nondeterministic Turing machine with polynomially bounded advice [6]. It is known to coincide with those languages whose projections are accepted by a sequence of nbps of size polynomial in n [9, 1]. Letting $\mathsf{nfa}(poly)$ contain those languages whose projections are accepted by a sequence of nfas of size polynomial in n, our main result is as follows:

Theorem. *$L \in \mathsf{NL}/poly$ iff there exists $k \in \mathbb{N}$ such that $n^k * L \in \mathsf{nfa}(poly)$.*

As we explain in the final section, one can also deduce other connections between complexity theory and automata theory:

(1) For any language $L \subseteq \mathbf{2}^*$ with poly-size *deterministic* branching programs, there exists $k \in \mathbb{N}$ such that $n^k * L$ is accepted by a sequence of poly-size nfas, each of which is a disjoint union of dfas.

(2) There is an explicit polynomial translation from propositional formulae ϕ to nfas N_ϕ, such that ϕ is a tautology iff N_ϕ accepts the full language $\mathbf{2}^*$.

These results illustrate the difficulty of minimising or proving lower bounds for nondeterministic finite automata. Although it is well known that minimising nfas is PSPACE-complete, our results imply hardness for many *specific* languages. For example, to show $L \notin \mathsf{NL}/poly$ it is necessary and sufficient to prove that, for each fixed natural k, the language $n^k * L$ does not have poly-size nfas. Showing this for any language $L \in \mathsf{NP}$ would prove that $\mathsf{NL} \neq \mathsf{NP}$. Also, by the contrapositive of (1) above, one could prove $\mathsf{L} \neq \mathsf{NL}$ by starting with some $L \in \mathsf{NL}$.

We discuss some connections with existing work. It is known that $\mathsf{NL}/poly = \mathsf{UL}/poly$ i.e. in some precise sense one can efficiently make nbps unambiguous [10]. However, our translation from nbps to nfas does not preserve unambiguity. This is unsurprising, since there exist good general methods for proving lower bounds for unambiguous nfas, essentially by computing the minimal dimension of any $\mathbb{Z}_2$-weighted machine accepting $L \subseteq \mathbf{2}^n$, viewed as a weighted language $L : \mathbf{2}^* \to \mathbb{Z}_2$. We have also considered the (extended) fooling technique and its generalisation, namely biclique edge coverings and their associated dimension [2], which is strongly related to communication complexity [8, 3]. These methods appear to be unsuitable for the languages we consider, although this approach deserves further study. Finally, Jukna has written a note concerning our construction, in which he explains that all known lower bound methods for read-once nbps do not apply to the languages we consider [5, 4].

2 Nondeterministic Branching Programs

In this section we review nondeterministic branching programs, providing comparisons to related structures and a normal form. We first fix our notation.

Notation. Let $\mathbf{2} = \{0, 1\}$ be the booleans and $\mathbb{N} = \{0, 1, 2, \dots\}$ be the set of natural numbers. For any language $L \subseteq \mathbf{2}^*$ let $\widetilde{L} = \mathbf{2}^* \setminus L$ denote its complement, and for any finite language $L \subseteq \mathbf{2}^n$ write $\overline{L} = \mathbf{2}^n \setminus L$ for its relative complement. Given any word $w \in \mathbf{2}^n$ and $1 \leq i \leq n$ let $w_i \in \mathbf{2}$ be the i^{th} letter of w. For any $d \in \mathbb{N}$ let $w^d = w \cdots w$ be the d-fold composition. Finally, fix a set $X_n = \{x_1, \dots, x_n\}$ of n variables for each natural $n \in \mathbb{N}$.

Definition 1. (a) A *nondeterministic branching program* (*nbp*) over n variables is a quadruple $\mathcal{B} = (G, s, \theta, \tau)$ consisting of:

(i) a finite directed multigraph $G = (V, E)$;

(ii) a *source node* $s \in V$;

(iii) a *node labelling* i.e. a function $\theta : V \to X_n \cup \mathbf{2}$ where every node labelled with 0 or 1 is a sink (has out-degree 0);

(iv) an *edge labelling* i.e. a function $\tau : E \to \mathbf{2}$.

We use the notation $(u\|l) \xrightarrow{b} (v\|m)$ to indicate that node u has label l, node v has label m, and there is an edge from u to v with label b.

(b) A *deterministic branching program* (*dbp*) is an nbp whose variable-labelled nodes have out-degree 2, where one outgoing edge is labelled by 0, the other by 1.

(c) The *size* $s(\mathcal{B})$ of an nbp $\mathcal{B}$ is its number of nodes. For acyclic $\mathcal{B}$, its *depth* $d(\mathcal{B})$ is the number of edges of any longest directed path starting at the source.

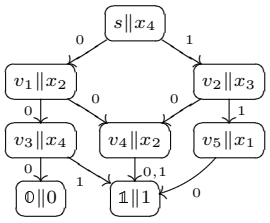
(d) A word $w = w_1 \dots w_n \in \mathbf{2}^n$ is *accepted* by an nbp $\mathcal{B}$ if there exists some path:

$$(s\|x_{k_0}) \xrightarrow{b_0} (v_1\|x_{k_1}) \xrightarrow{b_1} \cdots \xrightarrow{b_{m-2}} (v_{m-1}\|x_{k_{m-1}}) \xrightarrow{b_{m-1}} (v_m\|1)$$

consistent with w, i.e. $b_i = w_{k_i}$ for every $0 \leq i < m$. The *language* $L_{\mathcal{B}} \subseteq \mathbf{2}^n$ of $\mathcal{B}$ is the set of all words accepted by $\mathcal{B}$.

Remark 2. Many authors make additional assumptions on the structure of (nondeterministic) branching programs, e.g. that they are acyclic and every node is reachable from the source. These restrictions emerge in Lemma 8.

Example 3. Here is an nbp $\mathcal{B} = (G, s, \theta, \tau)$ over $n = 4$ variables:



Then $\mathcal{B}$ accepts the language:

$$L_{\mathcal{B}} = \{0000, 1000, 0010, 1010, 0001, 0101, 1001, 1101, 0011, 0111\}$$

i.e. the satisfying assignments of $(\bar{x}_4 \wedge \bar{x}_2) \vee (x_4 \wedge (\bar{x}_3 \vee (x_3 \wedge \bar{x}_1)))$. For example, 1010 is accepted via the path $(s\|x_4) \xrightarrow{0} (v_1\|x_2) \xrightarrow{0} (v_4\|x_2) \xrightarrow{0} (\mathbb{1}\|1)$.

Remark 4. Nbps are closely related to *switching-and-rectifier networks* (*srn*s) [9]. An srn $\mathcal{S} = (G, s, t, \mu)$ over n variables is a finite directed multigraph $G = (V, E)$ equipped with two vertices $s, t \in V$ (source and target) and a partial edge-labelling $\mu : E \rightharpoonup X_n \times \mathbf{2}$. A word $w \in \mathbf{2}^n$ is accepted iff there exists a directed path from s to t such that for each label (x_i, b) we have $w_i = b$. Define the *size* $s(\mathcal{S})$ of an srn $\mathcal{S}$ to be the number of nodes, although it is more standard to consider the number of *labelled edges*.

Every nbp $\mathcal{B}$ has an equivalent srn $\mathcal{S}$ with $s(\mathcal{S}) \leq s(\mathcal{B}) + 1$, and every srn $\mathcal{S}$ has an equivalent nbp $\mathcal{B}$ with $s(\mathcal{B}) \leq 1 + n \cdot s(\mathcal{S})$. By 'equivalent' we mean they accept the same language. The constructions resemble the translation between Moore and Mealy machines.

(a) Given an nbp $\mathcal{B} = (G, s, \theta, \tau)$ one can assume it has exactly one 1-labelled node $\mathbb{1}$ (else introduce a new 1-labelled node $\mathbb{1}$ and merge 1-labelled nodes). Then an equivalent srn $\mathcal{S}$ is obtained by labelling each edge (u, v) in G by $(\theta(u), \tau(u, v))$ and requiring $s/\mathbb{1}$ to be the source/target node, respectively. Therefore $s(\mathcal{S}) \leq 1 + s(\mathcal{B})$.

(b) Given any srn $\mathcal{S} = (G, s, t, \mu)$ over $n > 0$ variables, we may assume that:

(1) Any two labelled edges with the same source are labelled by the same variable x_i i.e. they have labels (x_i, b_j) for $j = 1, 2$. One can force this by adding unlabelled edges to 'dummy' nodes, used as the source of conflicting edges. At most $(n-1) \cdot s(\mathcal{S})$ new nodes are required.

(2) t is a sink, else add a new target t' and an unlabelled edge $t \to t'$.

(3) All edges are labelled, else replace every unlabelled edge (u, v) by two parallel edges (u, v), one labelled by $(x_i, 0)$ and one labelled by $(x_i, 1)$, where i is chosen such that (1) still holds.

Then $\mathcal{S}$ and $\mathcal{S}'$ accept the same language and $s(\mathcal{S}') \leq 1 + n \cdot s(\mathcal{S})$. We obtain $\mathcal{B}$ from $\mathcal{S}'$ as follows. Replace each (x_i, b)-labelled edge (u, v) by the b-labelled edge (u, v) and set $\theta(u) = x_i$ (well-defined by (1)). Finally label $\theta(t) = 1$ and $\theta(v) = 0$ for each sink $v \neq t$, and let s be the source of $\mathcal{B}$. Then $\mathcal{B}$ accepts the same language as $\mathcal{S}$, and $s(\mathcal{B}) = s(\mathcal{S}') \leq 1 + n \cdot s(\mathcal{S})$.

We now define a 'normal form' for nondeterministic branching programs.

Definition 5. An nbp $\mathcal{B} = (G, s, \theta, \tau)$ is called *stratified* if

(1) for any pair $e \neq e'$ of parallel edges, one has $\tau(e) \neq \tau(e')$;
(2) G is acyclic;
(3) every node is reachable from s;
(4) all sinks are labelled by 0 or 1;
(5) every path from s to a sink has length $d(\mathcal{B})$.

Remark 6. Assuming that (3) holds, the conditions (2) and (5) are equivalent to the existence of a (necessarily unique) partition $V = V_0 \cup V_1 \cup \ldots \cup V_{d(\mathcal{B})}$ such that $V_0 = \{s\}$, all sinks are contained in $V_{d(\mathcal{B})}$, and every edge of $\mathcal{B}$ goes from V_i to V_{i+1} for some $0 \leq i < d(\mathcal{B})$. In fact, choose V_i to be the nodes reachable from s via a path of length i.

Example 7. The nbp in Example 3 is stratified.

Lemma 8. Every nbp $\mathcal{B}$ has an equivalent stratified nbp of size $O(s(\mathcal{B})^6)$.

Proof. For $1 \leq k \leq 5$, we show that every nbp $\mathcal{B} = ((V, E), s, \theta, \tau)$ satisfying the first $k-1$ conditions of Definition 5 can be turned into an equivalent nbp satisfying the first k conditions.

$\underline{k = 1}$: Whenever an nbp $\mathcal{B}$ has parallel edges with the same label, delete all but one of them. This yields an equivalent nbp satisfying (1).

$\underline{k=2}$: If $\mathcal{B}$ satisfies (1), construct the nbp $\mathcal{B}' = ((V', E'), s', \theta', \tau')$ where:

$$V' = V \times \{0, \ldots, s(\mathcal{B})\} \qquad E' = \{((u,i),(v,i+1)) : 0 \leq i < s(\mathcal{B}),\ (u,v) \in E\}$$
$$s' = (s,0) \qquad \theta'(v,i) = \theta(v) \qquad \tau'((u,i),(v,i+1)) = \tau(u,v)$$

Clearly $\mathcal{B}'$ satisfies (1) and (2). Furthermore $\mathcal{B}'$ is equivalent to $\mathcal{B}$: if $w \in L_{\mathcal{B}}$ then there exists a w-consistent path $(s \| x_{k_0}) \xrightarrow{b_0} (v_1 \| x_{k_1}) \xrightarrow{b_1} \cdots \xrightarrow{b_{m-1}} (v_m \| 1)$ in $\mathcal{B}$ of length $m \leq s(\mathcal{B})$, which immediately yields the w-consistent path:

$$((s,0) \| x_{k_0}) \xrightarrow{b_0} ((v_1,1) \| x_{k_1}) \xrightarrow{b_1} \cdots \xrightarrow{b_{m-1}} ((v_m,m) \| 1)$$

in $\mathcal{B}'$. This shows $L_{\mathcal{B}} \subseteq L_{\mathcal{B}'}$ and the reverse inclusion is proved analogously.

$\underline{k=3}$: Given an nbp satisfying (1) and (2), restrict to those nodes reachable from the source via directed paths.

$\underline{k=4}$: Now assume that $\mathcal{B}$ satisfies (1)-(3). Then relabelling all variable-labelled sinks with 0 yields an equivalent nbp satisfying (1)-(4).

$\underline{k=5}$: We may assume that every sink is reachable from s via a path of length $d(\mathcal{B})$: Otherwise merge sinks with the same label, so that the resulting nbp has at most two sinks, and if there is sink of depth $i < d(\mathcal{B})$, extend it by a dummy path of length $d(\mathcal{B}) - i$. In view of Remark 6, define the partition:

$$V_i = \{v \in V : i \text{ is the length of any longest directed path from } s \text{ to } v\}$$

for each $0 \leq i \leq d(\mathcal{B})$. Clearly $V_0 = \{s\}$ and $V_{d(\mathcal{B})}$ contains all sinks. Furthermore, every edge goes from V_i to V_j for some $i < j$. By replacing any such edge by a $0,1$-labelled path of length $j - i$, one makes sure that every edge goes from V_i' to V_{i+1}' for some $i < d(\mathcal{B})$. By Remark 6, the resulting nbp satisfies (1)-(5).

Observe that in steps 2 and 5, the size of the constructed nbp is at most quadratic and cubic, respectively, in the size of the given one. In the other steps the size does not increase. Therefore, starting from any nbp $\mathcal{B}$ we have shown how to construct an equivalent stratified nbp of size $O(s(\mathcal{B})^6)$. □

3 From Stratified Nbps to Nfas

In this section we associate to each stratified nbp $\mathcal{B}$ a nondeterministic finite automaton $N_{\mathcal{B}}$ of size polynomial in $s(\mathcal{B})$. Although $N_{\mathcal{B}}$ does not accept the same language as $\mathcal{B}$, they are closely related. We start by recalling the classical notion of a nondeterministic finite automaton.

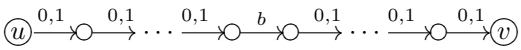
Definition 9. A *nondeterministic finite automaton* (*nfa*) is a tuple

$$N = (Z, (R_b)_{b=0,1}, F, I)$$

where Z is a finite set of states, $R_b \subseteq Z \times Z$ is a relation representing b-transitions, $F \subseteq Z$ is the set of final states, and $I \subseteq Z$ is the set of initial states. The *size* $s(N)$ of an nfa N is the number of states, and the *depth* $d(N)$ of N is the length of any longest path starting in an initial state (defined for acyclic nfas). N *accepts* the language $L(N) \subseteq \mathbf{2}^*$ in the usual manner: $w \in L(N)$ iff there exists a w-path from some initial state to some final state.

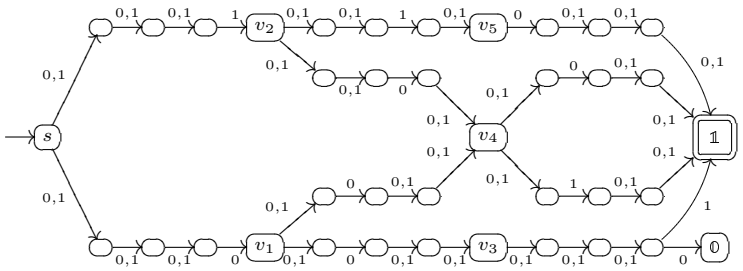
Remark 10. In analogy to Definition 5, we call an nfa N *stratified* if (i) N is acyclic and reachable, (ii) N has exactly one initial state, (iii) a state is final iff it is a sink, and (iv) all paths from the initial state to a final state have the same length $d(N)$. It is easy to see that every nfa accepting a finite language $L \subseteq \mathbf{2}^n$ can be turned into an equivalent stratified nfa with no more states. Moreover, a stratified nfa can be viewed as a stratified nbp: label final states with 1, label any nonfinal state with x_{i+1} if it is reachable via any word of length i, and choose the initial state as the source node. The resulting nbp is an instance of an *oblivious read-once* nbp: all paths from the source to a sink read each variable exactly once and in the same order.

Definition 11. Given a stratified nbp $\mathcal{B} = (G, s, \theta, \tau)$ over n variables, the nfa $N_{\mathcal{B}}$ is constructed as follows:
(1) Replace every edge $(u \| x_i) \xrightarrow{b} (v \| l)$ of $\mathcal{B}$ by a path of length n from u to v, where the i-th transition has label b and all others have labels 0 and 1.



(2) The source s is the only initial state of $N_{\mathcal{B}}$, and a state is final if and only if it is labelled with 1.

Example 12. For the nbp $\mathcal{B}$ of Example 3 we obtain the following nfa $N_{\mathcal{B}}$:



Lemma 15 below describes various relevant properties of $N_{\mathcal{B}}$. First we need a simple definition.

Definition 13. For each $n, d \in \mathbb{N}$ and finite language $L \subseteq \mathbf{2}^n$ define:

$$d \cdot L := \{w^d : w \in L\} \subseteq \mathbf{2}^{nd},$$

i.e. we take the collection of all d-powers of words from L.

Remark 14. It follows that $\overline{d \cdot L} = d \cdot \overline{L} \cup \overline{d \cdot \mathbf{2}^n}$ for any $n, d \geq 0$ and $L \subseteq \mathbf{2}^n$. That is, this relative complement consists of (a) those d-powered words w^d where $w \notin L$, and (b) those words in $\mathbf{2}^{nd}$ which are not d-powers.

Lemma 15. For any stratified nbp $\mathcal{B}$ over n variables, we have:

(a) $s(N_\mathcal{B}) = O(n \cdot s(\mathcal{B})^2)$, $d(N_\mathcal{B}) = n \cdot d(\mathcal{B})$ and $L(N_\mathcal{B}) \subseteq \mathbf{2}^{n \cdot d(\mathcal{B})}$.

(b) $d(\mathcal{B}) \cdot L_\mathcal{B} = L(N_\mathcal{B}) \cap (d(\mathcal{B}) \cdot \mathbf{2}^n)$. That is, the $d(\mathcal{B})$-powers of words accepted by $\mathcal{B}$ are precisely those $d(\mathcal{B})$-powered words that $N_\mathcal{B}$ accepts.

Proof. (a) follows directly from the construction of $N_\mathcal{B}$.
(b) Let $d = d(\mathcal{B})$. To prove '$\subseteq$', suppose $w \in L_\mathcal{B}$, so there exists some path:

$$s = (v_0 \| x_{k_0}) \xrightarrow{b_0} (v_1 \| x_{k_1}) \xrightarrow{b_1} \ldots \xrightarrow{b_{d-1}} (\mathbb{1} \| 1) \qquad (*)$$

in $\mathcal{B}$ with $b_i = w_{k_i}$ for all i. This yields accepting paths of the form

$$(s = v_0 \xrightarrow{c_{0,1}} \ldots \xrightarrow{c_{0,n}})(v_1 \xrightarrow{c_{1,1}} \ldots \xrightarrow{c_{1,n}}) \ldots (v_{d-1} \xrightarrow{c_{d-1,0}} \ldots \xrightarrow{c_{d-1,n}} \mathbb{1}) \qquad (**)$$

in $N_\mathcal{B}$ where $c_{i,j} = b_i$ for $j = k_i$, and $c_{i,j} \in \mathbf{2}$ is arbitrary otherwise. In particular, choosing $c_{i,j} = w_j$ for all i and j yields an accepting path for the word w^d. Hence $w^d \in L(N_\mathcal{B}) \cap d \cdot \mathbf{2}^n$.

Conversely, any accepting path in $N_\mathcal{B}$ is induced by some path $(*)$ in $\mathcal{B}$ and has the form $(**)$. If a word w^d ($w \in \mathbf{2}^n$) is accepted in $N_\mathcal{B}$ via $(**)$, we have $b_i = c_{i,k_i} = w_{k_i}$ for all i, so the path $(*)$ in $\mathcal{B}$ is consistent with w. It follows that $w \in L_\mathcal{B}$, which proves '$\supseteq$'. □

4 Characterisation of NL/*poly*

We are now ready to prove our characterisation of NL/*poly* via nondeterministic finite automata. We first introduce the relevant complexity classes.

Notation. For any language $L \subseteq \mathbf{2}^*$ and $n \in \mathbb{N}$, let $L_n := L \cap \mathbf{2}^n$.

Definition 16. The complexity class nbp(*poly*) contains those $L \subseteq \mathbf{2}^*$ such that each L_n is accepted by some nbp $\mathcal{B}_n$, where $s(\mathcal{B}_n) \in n^{O(1)}$ i.e. their size is bounded polynomially in n. The complexity classes dbp(*poly*) and nfa(*poly*) are defined analogously: replace 'nbp' by 'dbp' or 'nfa', respectively.

The following relationships are well-known:

$$\mathsf{dbp}(poly) = \mathsf{L}/poly \qquad \mathsf{nbp}(poly) = \mathsf{NL}/poly$$

where L/*poly* (resp. NL/*poly*) consists of those languages accepted by some single log-space bounded deterministic (resp. nondeterministic) Turing machine with polynomially bounded advice [6]. These results are mentioned in [9], where our dbps correspond to 'BPs' and their notion of size agrees up to a linear factor.

On the other hand, although our nbps are not quite the same as the switching-and-rectifier networks used in [9] (their size is the number of labelled edges), the above correspondence nevertheless holds, see [1, Theorem 1].

For any language $L \subseteq \mathbf{2}^*$ and function $t : \mathbb{N} \to \mathbb{N}$, define:

$$t * L := \bigcup_{n \geq 0} \overline{t(n) \cdot L_n} \ \subseteq \ \mathbf{2}^*$$

Recall that $\overline{t(n) \cdot L_n} \subseteq \mathbf{2}^{n \cdot t(n)}$ is the relative complement of $t(n) \cdot L_n$, the latter being the $t(n)$-powers of words in L_n (see Definition 13).

Theorem 17. $L \in \mathsf{nbp}(poly)$ iff there exists $k \in \mathbb{N}$ such that $n^k * L \in \mathsf{nfa}(poly)$.

The proof uses the following two results. The first is a corollary of the Immerman-Szelepcsényi theorem, as mentioned in [9].

Theorem 18. The class $\mathsf{nbp}(poly)$ is closed under complement:

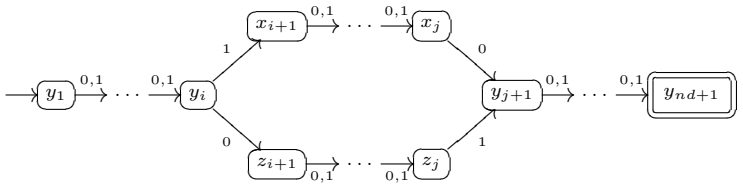
$$L \in \mathsf{nbp}(poly) \quad \text{iff} \quad \widetilde{L} \in \mathsf{nbp}(poly)$$

for any language $L \subseteq \mathbf{2}^*$.

The second result provides poly-size nfas for certain finite languages.

Lemma 19. For all $n, d \in \mathbb{N}$, there exists an nfa with $O(n^2 d^3)$ states accepting the language $\overline{d \cdot \mathbf{2}^n} \subseteq \mathbf{2}^{nd}$.

Proof. $\overline{d \cdot \mathbf{2}^n}$ consists of all words $w \in \mathbf{2}^{nd}$ such that there exists $1 \leq i < j \leq n \cdot d$ where (i) $i = j \bmod n$, and (ii) $w_i \neq w_j$. The following nfa with $O(nd)$ states accepts all such words for a fixed pair (i, j):



Taking the disjoint union of these nfas yields an nfa accepting $\overline{d \cdot \mathbf{2}^n}$. Since there are $n\binom{d}{2} = O(nd^2)$ pairs (i, j) satisfying (i), this nfa has $O(n^2 d^3)$ states. □

Remark 20. On the other hand, poly-size nfas do *not* exist for the relative complements $d \cdot \mathbf{2}^n \subseteq \mathbf{2}^{nd}$. In fact, a state-minimal nfa for $d \cdot \mathbf{2}^n$ is obtained from its state-minimal dfa by deleting the state accepting the empty language. The latter is exponential in n for any fixed $d > 1$. To see this, one can use the fact that $d \cdot \mathbf{2}^n$ defines a *linear code* i.e. a linear subspace of $\mathbb{Z}_2^{nd}$.

Proof (Theorem 17). Let $L \in \mathsf{nbp}(poly)$. Then also $\widetilde{L} \in \mathsf{nbp}(poly)$ by Theorem 18, so there exists a family of nbps $\mathcal{B}_n$ ($n \geq 0$) such that $\mathcal{B}_n$ accepts $(\widetilde{L})_n = \overline{L_n}$ and $s_n := s(\mathcal{B}_n)$ is polynomially bounded in n. By Lemma 8, we may assume that the nbps $\mathcal{B}_n$ are stratified. Moreover, we assume that $d_n := d(\mathcal{B}_n) = n^k$ for some $k \in \mathbb{N}$ (otherwise add dummy paths). Let $N_n := N_{\mathcal{B}_n}$ be the nfa associated to $\mathcal{B}_n$ (see Definition 11). Then:

$$\begin{aligned}
\overline{d_n \cdot L_n} &= d_n \cdot \overline{L_n} \cup \overline{d_n \cdot \mathbf{2}^n} && \text{see Remark 14}\\
&= d_n \cdot L_{\mathcal{B}_n} \cup \overline{d_n \cdot \mathbf{2}^n} && \text{since } L_{\mathcal{B}_n} = \overline{L_n}\\
&= (L(N_n) \cap d_n \cdot \mathbf{2}^n) \cup \overline{d_n \cdot \mathbf{2}^n} && \text{by Lemma 15.}(b)\\
&= (L(N_n) \cup \overline{d_n \cdot \mathbf{2}^n}) \cap (d_n \cdot \mathbf{2}^n \cup \overline{d_n \cdot \mathbf{2}^n})\\
&= L(N_n) \cup \overline{d_n \cdot \mathbf{2}^n}
\end{aligned}$$

By Lemma 15, N_n has $O(ns_n^2)$ states. Moreover, by Lemma 19 there exists an nfa N'_n accepting $\overline{d_n \cdot \mathbf{2}^n}$ with $O(n^2 d_n^3)$ states, this being polynomial in n because $d_n \leq s_n$. Taking the disjoint union of the nfas N_n and N'_n yields a polynomial-sized nfa N''_n for $L(N_n) \cup \overline{d_n \cdot \mathbf{2}^n} = \overline{d_n \cdot L_n}$.

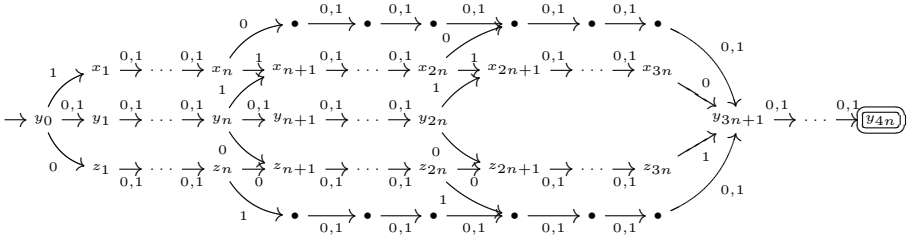
Then we obtain a polynomial-sized family of nfas M_m accepting $(d_n * L)_m$ as follows. If $m = nd_n (= n^{k+1})$ for some n, we have $(d_n * L)_m = \overline{d_n \cdot L_n}$, so take $M_m = N''_n$. The size of N''_n is polynomial in n, hence also in $m = n^{k+1}$. If m is not of the form $nd_n = n^{k+1}$ for some n then $(d_n * L)_m = \emptyset$, so let M_m be a one-state nfa accepting $\emptyset$. This proves $d_n * L \in \mathsf{nfa}(poly)$.

For the converse, suppose we have $n^k * L \in \mathsf{nfa}(poly)$ for some $k \in \mathbb{N}$. Then there exists a family of polynomial-sized nfas N_m ($m \in \mathbb{N}$) accepting $(n^k * L)_m$. By Remark 10, we can turn N_m into an equivalent stratified (oblivious read-once) nbp $\mathcal{B}_m$ of the same size. Then by Theorem 18, there also exists a family of polynomial-sized nbps $\mathcal{B}'_m$ accepting $(\widetilde{n^k * L})_m$. If $m = n^{k+1}$ for some n, then $(\widetilde{n^k * L})_m = n^k \cdot L_n$ and the size of $\mathcal{B}'_m$ is polynomial in n, since it is polynomial in $m = n^{k+1}$. The nbp $\mathcal{B}'_m$ has n^{k+1} variables $x_1, \ldots, x_{n^{k+1}}$, and replacing all node labels $x_{p \cdot n+i}$ (where $0 \leq p < n^k$ and $1 \leq i \leq n$) by x_i yields an nbp $\mathcal{B}''_n$ accepting L_n whose size is polynomial in n. It follows that $L \in \mathsf{nbp}(poly)$. □

5 Applications

It immediately follows that $L \in \mathsf{NL}/poly$ iff there exists $k \in \mathbb{N}$ such that $n^k * L$ has poly-size nfas. Equivalently, to show L does *not* lie in $\mathsf{NL}/poly$ it is necessary and sufficient to prove that, for each fixed k, any sequence of nfas accepting $n^k * L$'s projections have size super-polynomial in n. Proving this for some $L \in \mathsf{NP}$ would imply $\mathsf{NL} \neq \mathsf{NP}$. Furthermore, if some $n^k * L$ *did* have poly-size nfas, then L has non-uniform poly-size boolean circuits: (i) view the nfas as acyclic nbps and linearly translate to boolean circuits, (ii) identify variables $x_i = x_j$ whenever $|j - i| = 1 \bmod n$, (iii) add a NOT gate to the output. In particular, either an NP-complete language L has non-uniform poly-size circuits, or for each k the language $n^k * L$ does not have poly-size nfas.

Next we show that the non-powers $\overline{d \cdot \mathbf{2}^n} \subseteq \mathbf{2}^{nd}$ are accepted by an nfa of size $O(n^2 d)$, improving the $O(n^2 d^3)$ bound in Lemma 19. For each $1 \leq i \leq n$, there is an nfa N_i of size $O(nd)$ accepting those $w \in \mathbf{2}^{nd}$ such that $w_{nx+i} \neq w_{ny+i}$ for some $1 \leq x, y < d$. Below we have drawn N_1 in the case where $d = 4$.



The disjoint union of the N_i's accepts $\overline{d \cdot \mathbf{2}^n}$ and has size $O(n^2 d)$. However the N_i's are inherently nondeterministic, whereas the nondeterministic acceptor described in Lemma 19 is a disjoint union of partial dfas, which turns out to be useful. First note that dbps are nbps, so for any $L \in \mathsf{L}/poly$ there is some $n^k * L$ with poly-size nfas. Then one can strengthen this as follows:

Lemma 21. If $L \in \mathsf{L}/poly$ then there exists $k \in \mathbb{N}$ such that $n^k * L$ has poly-size nfas, each of which is a disjoint union of dfas.

Proof. If $L \in \mathsf{L}/poly$ there exist poly-size dbps $\mathcal{B}_n$ of depth d_n accepting L_n. We may assume they are stratified with $d_n = n^k$ for some $k \in \mathbb{N}$, and construct the associated nfas $N_{\mathcal{B}_n}$. Since $\mathcal{B}_n$ is a dbp, $N_{\mathcal{B}_n}$ is very nearly a *partial dfa*: some nondeterminism arises via parallel $0,1$-edges from the dbp but it may easily be eliminated by identifying states, see node v_4 in Example 12. Then we can construct dfas D_n accepting $\overline{L(N_{\mathcal{B}_n})}$ of essentially the same size, so put them in parallel with the dfas from Lemma 19 to obtain nfas N_n accepting:

$$L(N_n) = L(D_n) \cup \overline{d_n \cdot \mathbf{2}^n} = \overline{L(N_{\mathcal{B}_n})} \cup \overline{d_n \cdot \mathbf{2}^n} = \overline{d_n \cdot L(\mathcal{B}_n)} = \overline{d_n \cdot L_n}$$

using Lemma 15.(b) in the penultimate step. Thus $n^k * L$ has poly-size nfas N_n, each one being a disjoint union of dfas. □

Therefore to prove $\mathsf{L} \neq \mathsf{NL}$ it suffices to find some $L \in \mathsf{NL}$ such that, for each fixed $k \in \mathbb{N}$, the language $n^k * L$ does not have poly-size nfas of this form. This bares a resemblance to work on 2-dfas simulating nfas [11, 12, 7], where it is believed that certain nfas which somehow encode 'reachability' cannot be polynomially simulated by 2-dfas. However, there is a significant difference: the work on 2-dfas uses sequences of nfas accepting infinite regular languages, whereas we work with sequences of *finite* languages.

Finally we describe a polynomial translation between propositional formulae ϕ and nfas N_ϕ, such that ϕ is a tautology iff the nfa N_ϕ accepts the full language $\mathbf{2}^*$. We may assume the formula ϕ is in negation normal form and only mentions the variables $x_1, \ldots, x_n$. Then there is a linear translation from ϕ to an acyclic nbp $\mathcal{B}_\phi$ accepting ϕ's satisfying valuations $L_\phi \subseteq \mathbf{2}^n$ [4]. Briefly, variables x_i

and negated variables $\bar{x}_i$ become $x_i \xrightarrow{1} 1$ and $x_i \xrightarrow{0} 1$ respectively, whereas disjunctions/conjunctions become parallel/sequential composition respectively. Stratifying $\mathcal{B}_\phi$, one obtains an nbp $\mathcal{B}'_\phi$ accepting L_ϕ whose depth d_ϕ is the maximal nesting depth of conjunctions in ϕ (appropriately defined). Applying our translation, we obtain an nfa $N_{\mathcal{B}'_\phi}$ of depth nd_ϕ whose accepted d_ϕ-powered words are precisely $d_\phi \cdot L_\phi$. Finally, put this nfa in parallel with:

(i) The $O(n^2 d_\phi)$ sized nfa accepting the non-powers $\overline{d_\phi \cdot \mathbf{2}^n} \subseteq \mathbf{2}^{nd_\phi}$.
(ii) An $O(nd_\phi)$ sized nfa accepting $\mathbf{2}^* \setminus \mathbf{2}^{nd_\phi}$.

The resulting nfa N_ϕ accepts the full language $\mathbf{2}^*$ iff it accepts every d_ϕ-power in $\mathbf{2}^{nd_\phi}$ iff $L_\phi = \mathbf{2}^n$ iff ϕ is a tautology. Furthermore $s(N_\phi)$ is polynomial in $s(\mathcal{B}_\phi)$ and hence also in the size of ϕ, where one usually counts the leaves.

References

[1] Carsten Damm and Markus Holzer. Inductive counting below logspace. In *Mathematical Foundations of Computer Science 1994*, volume 841 of *Lecture Notes in Computer Science*, pages 276–285. Springer Berlin Heidelberg, 1994.

[2] Hermann Gruber and Markus Holzer. Finding lower bounds for nondeterministic state complexity is hard. In *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 363–374. Springer Berlin Heidelberg, 2006.

[3] Juraj Hromkovi, Juhani Karhumki, Hartmut Klauck, Georg Schnitger, and Sebastian Seibert. Measures of nondeterminism in finite automata. In *Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 199–210. Springer Berlin Heidelberg, 2000.

[4] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[5] Stasys Jukna. What have read-once branching programs to do with nl/poly? `http://www.thi.informatik.uni-frankfurt.de/~jukna/boolean/comments.html`, 2013. [Online, see Comment 12].

[6] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, STOC '80, pages 302–309, New York, NY, USA, 1980. ACM.

[7] Hing Leung. Tight lower bounds on the size of sweeping automata. *J. Comput. Syst. Sci.*, 63(3):384–393, 2001.

[8] M. Pal. Communication complexity and parallel computing. *IEEE Concurrency*, 7(2):81–83, 1999.

[9] Alexander A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *Proceedings of the 8th International Symposium on Fundamentals of Computation Theory*, FCT '91, pages 47–60. Springer-Verlag, 1991.

[10] K. Reinhardt and E. Allender. Making nondeterminism unambiguous. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 244–253, 1997.

[11] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *STOC*, pages 275–286. ACM, 1978.

[12] Michael Sipser. Lower bounds on the size of sweeping automata. *J. Comput. Syst. Sci.*, 21(2):195–202, 1980.