Electronic Colloquium on Computational Complexity, Report No. 75 (2019)

Relations and Equivalences Between Circuit Lower Bounds and Karp-Lipton Theorems*

Lijie Chen
MIT

Dylan M. McKay
MIT

Cody D. Murray†

R. Ryan Williams
MIT

Abstract

A frontier open problem in circuit complexity is to prove $\mathsf{P}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[n^k]$ for all k; this is a necessary intermediate step towards $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$. Previously, for several classes containing P^{NP}, including $\mathsf{NP}^{\mathsf{NP}}$, $\mathsf{ZPP}^{\mathsf{NP}}$, and $\mathsf{S}_2\mathsf{P}$, such lower bounds have been proved via *Karp-Lipton-style Theorems*: to prove $\mathcal{C} \not\subset \mathsf{SIZE}[n^k]$ for all k, we show that $\mathcal{C} \subset \mathsf{P}_{/\mathsf{poly}}$ implies a "collapse" $\mathcal{D} = \mathcal{C}$ for some larger class $\mathcal{D}$, where we already know $\mathcal{D} \not\subset \mathsf{SIZE}[n^k]$ for all k.

It seems obvious that one could take a different approach to prove circuit lower bounds for P^{NP} that does not require proving any Karp-Lipton-style theorems along the way. We show this intuition is *wrong*: **(weak) Karp-Lipton-style theorems for P^{NP} are *equivalent* to fixed-polynomial size circuit lower bounds for P^{NP}**. That is, $\mathsf{P}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[n^k]$ for all k if and only if ($\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$ implies $\mathsf{PH} \subset$ i.o.-$\mathsf{P}^{\mathsf{NP}}_{/n}$).

Next, we present new consequences of the assumption $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$, towards proving similar results for NP circuit lower bounds. We show that under the assumption, fixed-polynomial circuit lower bounds for NP, nondeterministic polynomial-time derandomizations, and various fixed-polynomial time simulations of NP are all *equivalent*. Applying this equivalence, we show that **circuit lower bounds for NP imply better Karp-Lipton collapses**. That is, if $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k, then for all $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PP}, \mathsf{PSPACE}, \mathsf{EXP}\}$, $\mathcal{C} \subset \mathsf{P}_{/\mathsf{poly}}$ implies $\mathcal{C} \subset$ i.o.-$\mathsf{NP}_{/n^{\varepsilon}}$ for all $\varepsilon > 0$. Note that unconditionally, the collapses are only to MA and not NP.

We also explore consequences of circuit lower bounds for a *sparse* language in NP. Among other results, we show if a polynomially-sparse NP language does not have $n^{1+\varepsilon}$-size circuits, then $\mathsf{MA} \subset$ i.o.-$\mathsf{NP}_{/O(\log n)}$, $\mathsf{MA} \subset$ i.o.-$\mathsf{P}^{\mathsf{NP}[O(\log n)]}$, and $\mathsf{NEXP} \not\subset \mathsf{SIZE}[2^{o(m)}]$. Finally, we observe connections between these results and the "hardness magnification" phenomena described in recent works.

1 Introduction

Let $\mathcal{C}$ be a complexity class containing NP. A longstanding method for proving fixed-polynomial circuit lower bounds for functions in $\mathcal{C}$, first observed by Kannan [Kan82], applies versions of the classical Karp-Lipton Theorem in a particular way:

1. If $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$, then $\mathsf{SAT} \in \mathsf{NP} \subset \mathcal{C}$ does not have polynomial-size circuits.
2. If $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$, then by a "collapse" theorem, we have $\mathsf{PH} \subseteq \mathcal{C}$. But for every k, there is an $f \in \mathsf{PH}$ that does not have n^k-size circuits, so we are also done.

Such collapse theorems are called *Karp-Lipton Theorems*, as they were first discovered by Karp and Lipton [KL82] in their pioneering work on complexity classes with advice. The general theme of such theorems is a connection between non-uniform and uniform complexity:

*Supported by NSF CCF-1741615 (CAREER: Common Links in Algorithms and Complexity).
†Work done while the author was a PhD student at MIT.

ISSN 1433-8092

"$\mathcal{C}$ has (non-uniform) polynomial-size circuits implies a collapse of (uniform) complexity classes."

Over the years, Karp-Lipton Theorems have been applied to prove circuit lower bounds for the complexity classes $\mathsf{NP}^{\mathsf{NP}}$ [Kan82], $\mathsf{ZPP}^{\mathsf{NP}}$ [BCG+96, KW98], $\mathsf{S_2P}$ [Cai07, CCHO05], PP [Vin05, Aar06][1], and Promise-MA and $\mathsf{MA}/1$ [San09].[2] Other literature on Karp-Lipton Theorems include [Yap83, CR06, CR11].

When one first encounters such a lower bound argument, the non-constructivity of the result (the two uncertain cases) and the use of a Karp-Lipton Theorem looks strange.[3] It appears obvious that one ought to be able to prove circuit lower bounds in a fundamentally *different way*, without worrying over any collapses of the polynomial hierarchy. It is easy to imagine the possibility of a sophisticated combinatorial argument establishing a lower bound for P^{NP} functions (one natural next step in such lower bounds) which has nothing to do with simulating PH more efficiently, and has no implications for it.

P^{NP} Circuit Lower Bounds are Equivalent to Karp-Lipton Collapses to P^{NP}. We show that, in a sense, the above intuition is **false**: any fixed-polynomial-size circuit lower bound for P^{NP} would imply a Karp-Lipton Theorem collapsing PH all the way to P^{NP}. (There are some technicalities: the P^{NP} simulation uses small advice and only works infinitely often, but we believe these conditions can potentially be removed, and they do not change the moral of our story.) We find this result surprising; it shows that *in order to prove a circuit lower bound for P^{NP}, one cannot avoid proving a Karp-Lipton Theorem for P^{NP} in the process*. A Karp-Lipton Theorem is both necessary and sufficient for such lower bounds.

Theorem 1.1 (P^{NP} Circuit Lower Bounds are Equivalent to a Karp-Lipton Collapse to P^{NP})**.**
$\mathsf{P}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[n^k]$ for all k if and only if ($\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{PH} \subset$ i.o.-$\mathsf{P}^{\mathsf{NP}}_{/n}$).

One direction of Theorem 1.1 follows immediately from the classical lower bound paradigm described above. In particular, assuming $\mathsf{P}^{\mathsf{NP}} \subset \mathsf{SIZE}[n^k]$ for some k and assuming $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{PH} \subset$ i.o.-$\mathsf{P}^{\mathsf{NP}}_{/n}$ we have

$$\mathsf{PH} \subset \text{i.o.-}\mathsf{P}^{\mathsf{NP}}_{/n} \subseteq \text{i.o.-}\mathsf{SIZE}[O(n)^k],$$

which contradicts known fixed-polynomial lower bounds for PH. The interesting direction is the converse, showing that *proving lower bounds against P^{NP}* **implies** *proving a Karp-Lipton collapse to P^{NP} that is sufficient for the lower bound.*

NP Circuit Lower Bounds Imply Better Karp-Lipton Collapses. After observing Theorem 1.1, a natural question is whether such a theorem holds for NP circuit lower bounds as well:

Does $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k imply a Karp-Lipton Collapse to NP?

While we have not yet been able to prove this under the hypothesis $\mathsf{NP} \subset \mathsf{P}/\operatorname{poly}$ as above, we can show it for stronger hypotheses. Another class of Karp-Lipton Theorems (used in circuit lower bounds for PP [Vin05, Aar06] and Promise-MA [San09]) give stronger collapses under hypotheses like $\mathsf{PSPACE} \subset \mathsf{P}/\operatorname{poly}$: for any class $\mathcal{C}$ which is one of NEXP [IKW02], $\mathsf{EXP}^{\mathsf{NP}}$ ([BH92] and [BFLS91]), EXP and PSPACE [BFLS91], PP [LFKN92] and $\oplus\mathsf{P}$ [IKV18], we have:

If $\mathcal{C} \subset \mathsf{P}/\operatorname{poly}$ then $\mathcal{C} \subseteq \mathsf{MA}$.

[1] Both Vinodchandran and Aaronson's proofs of $\mathsf{PP} \not\subset \mathsf{SIZE}[n^k]$ use the Karp-Lipton-style theorem "$\mathsf{PP} \subset \mathsf{P}/\operatorname{poly}$ then $\mathsf{PP} = \mathsf{MA}$", which follows from [LFKN92]. Aaronson shows further that "$\mathsf{PP} \subset \mathsf{P}/\operatorname{poly}$ then $\mathsf{P}^{\mathsf{PP}} = \mathsf{MA}$". From there, one can directly construct a function in P^{PP} without n^k-size circuits.

[2] Santhanam used the Karp-Lipton-style theorem "$\mathsf{PSPACE} \subset \mathsf{P}_{/\mathsf{poly}}$ implies $\mathsf{PSPACE} = \mathsf{MA}$" to prove lower bounds against Promise-MA and MA with one bit of advice.

[3] Note Cai and Watanabe [CW04] found a constructive proof for $\mathsf{NP}^{\mathsf{NP}}$.

We show how NP circuit lower bounds can be used to derandomize MA. In fact, under the hypothesis $\mathsf{NP} \subset \mathsf{P}/\operatorname{poly}$, we prove an equivalence between NP circuit lower bounds, fast Arthur-Merlin simulations of NP, and nondeterministic derandomization of Arthur-Merlin protocols.

To state our results, we first define a variation of the "robust simulation" which was originally introduced in [FS17]. For a complexity class $\mathcal{C}$ and a language L, we say L is in c-r.o.-$\mathcal{C}$ for a constant c, if there is a language $L' \in \mathcal{C}$ such that there are infinitely many m's such that for all $n \in [m, m^c]$, L' agrees with L on inputs of length n.[4] (See Section 2.1 for formal definitions.)

Theorem 1.2. *Assuming $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$, the following are equivalent:*

1. *NP is not in $\mathsf{SIZE}[n^k]$ for all k.*
2. *$\mathsf{AM}_{/1}$ is in c-r.o.-$\mathsf{NP}_{/n^\varepsilon}$ for all $\varepsilon > 0$ and integers c.*
   That is, Arthur-Merlin games with $O(1)$ rounds and small advice can be simulated "c-robustly often" in NP with modest advice, for all constants c.[5]
3. *NP does not have n^k-size witnesses for all k.*
   That is, for all k, there is a language $L \in \mathsf{NP}$, a poly-time verifier V for L, and infinitely many $x_n \in L$ such that $V(x_n, \cdot)$ has no witness of circuit complexity at most n^k.
4. *For all k and d, there is a polynomial-time nondeterministic PRG with seed-length $O(\log n)$ and n bits of advice against n^k-size circuits d-robustly often.*[6]
5. *NP is not in $\mathsf{AMTIME}(n^k)$ for all k.*
6. *$(\mathsf{NP} \cap \mathsf{coNP})_{/n^\varepsilon}$ is not in $\mathsf{SIZE}[n^k]$ for all k and all $\varepsilon > 0$.*
7. *$(\mathsf{AM} \cap \mathsf{coAM})_{/1}$ is in c-r.o.-$(\mathsf{NP} \cap \mathsf{coNP})_{/n^\varepsilon}$ for all $\varepsilon > 0$ and all integers c.*

That is, under $\mathsf{NP} \subset \mathsf{P}/\operatorname{poly}$, the tasks of fixed-polynomial lower bounds for NP, lower bounds for $(\mathsf{NP} \cap \mathsf{coNP})/n^\varepsilon$, uniform lower bounds on simulating NP within AM, and derandomizing AM in NP are all equivalent.

We recall another type of Karp-Lipton collapse was shown by [AKSS95]: $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$ implies $\mathsf{AM} = \mathsf{MA}$. An intriguing corollary of Theorem 1.2 is that fixed-polynomial lower bounds for NP would improve this collapse, from MA to r.o.-c-$\mathsf{NP}_{/n^\varepsilon}$ for all c:

Corollary 1.3 (NP Circuit Lower Bounds Equivalent to a Karp-Lipton Collapse of AM to NP)**.** *$\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k if and only if ($\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{AM}$ is in r.o.-c-$\mathsf{NP}_{/n^\varepsilon}$ for all c).*

Another consequence of Theorem 1.2 is that NP circuit lower bounds imply better Karp-Lipton collapses from MA down to NP:

Theorem 1.4 (NP Circuit Lower Bounds Imply Better Karp-Lipton Collapses)**.** *Let $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PSPACE}, \mathsf{PP}, \mathsf{EXP}\}$. Suppose $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k. Then for all $\varepsilon > 0$, ($\mathcal{C} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathcal{C} \subset$ i.o.-$\mathsf{NP}_{/n^\varepsilon}$). In particular, polynomial-size circuits for any $\mathcal{C}$-complete language L can be constructed in NP infinitely often, with n^ε advice.*

Remark 1.5. *By "circuits for L can be constructed in NP infinitely often", we mean that there is a nondeterministic poly-time algorithm A such that, for infinitely many n, A on input 1^n outputs a circuit C_n for L_n on at least one computation path, and on all paths where such a C_n is not output, A outputs* reject.

[4] The original definition of $L \subseteq$ r.o.-$\mathcal{C}$ requires that there is a *single* language $L' \in \mathcal{C}$ such that for all c there are infinitely many m's such that for all $n \in [m, m^c]$, L' agrees with L on inputs of length n.

[5] See the Preliminaries for a definition of "c-robustly often". Intutively, it is a mild strengthening of "infinitely often".

[6] See the Preliminaries for formal definitions.

Consequences of Weak Circuit Lower Bounds for Sparse Languages in NP. Theorem 1.2 shows that assuming $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$, fixed-polynomial lower bounds for NP imply $\mathsf{AM} = \mathsf{MA} \subseteq$ i.o.-$\mathsf{NP}_{/n^{\varepsilon}}$. This is also the reason that we can only show collapses to i.o.-$\mathsf{NP}_{/n^{\varepsilon}}$ in Theorem 1.4. It is interesting to ask whether the n^{ε} advice in the simulation can be eliminated or reduced. In the following, we show that an $n^{1.00001}$-size circuit lower bound for a polynomially-sparse language in NP would imply an advice reduction, along with other interesting consequences.

Theorem 1.6 (Consequences of Weak Circuit Lower Bounds for Polynomially-Sparse NP Languages)**.** *Suppose there is an $\varepsilon > 0$, a $c \geq 1$, and an n^c-sparse $L \in \mathsf{NP}$ without $n^{1+\varepsilon}$-size circuits. Then $\mathsf{MA} \subset$ i.o.-$\mathsf{NP}_{/O(\log n)}$, $\mathsf{MA} \subseteq$ i.o.-$\mathsf{P}^{\mathsf{NP}[O(\log n)]}$, and $\mathsf{NE} \not\subset \mathsf{SIZE}[2^{\delta \cdot n}]$ for some $\delta > 0$ (which implies $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k).*

One step in the proof of Theorem 1.6 is a form of *hardness condensation* (as termed by Impagliazzo [Imp18]) for sparse NP languages. The goal of hardness condensation [BS06, IJKW10] is that, given a function f on n input bits with complexity S, we want to construct a function $\widetilde{f}$ on $\ell \ll n$ input bits that still has complexity roughly S. We show how a hard $S(n)$-sparse language in $\mathsf{NTIME}[T(n)]$ can be "condensed" in a generic way, based on the sparsity $S(n)$. We can efficiently build a PRG from the harder condensed function.

Theorem 1.6 shows how a very weak lower bound ($n^{1+\varepsilon}$) for a sparse language $L \in \mathsf{NP}$ would imply an *exponential-size* lower bound for NE (note, the converse is easy to show). This is reminiscent of a recent line of work [OS18, OPS19, MMW19] on "hardness magnification" phenomena, showing that seemingly weak circuit lower bounds for certain problems can in fact imply strong circuit lower bounds which are out of reach of current proof techniques.

At a high level, the hardness magnification results in the above-cited papers show how weak lower bounds on "compression problems" can imply strong complexity class separations. These compression problems have the form: *given a string, does it have a small efficient representation?* As an example, in the Minimum Circuit Size Problem for size $S(m) \ll 2^m$, denoted as $\mathsf{MCSP}[S(m)]$, we are given a truth table of length $N = 2^m$ and want to know if the function has a circuit of size at most $S(m)$. As an example of hardness magnification, McKay, Murray, and Williams [MMW19] show that, if there is an $\varepsilon > 0$ such that $\mathsf{MCSP}[2^{m/\log^{\star} m}]$ is not in $\mathsf{SIZE}[N^{1+\varepsilon}]$, then $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$. Thus a very weak circuit size lower bound for $\mathsf{MCSP}[2^{m/\log^{\star} m}]$ would imply a super-polynomial lower bound for SAT!

Sparsity Alone Implies a Weak Hardness Magnification. We identify a simple property of all efficient compression problems which alone implies a (weak) form of hardness magnification: the *sparsity of the underlying language*. For any compression problem on length-N strings where we ask for a length-$\ell(N)$ representation (think of $\ell(N) \leq n^{o(1)}$), there are at most $2^{\ell(N)}$ strings in the language. Scaling up the sparsity of Theorem 1.6, we show that non-trivial circuit lower bounds for *any* NP problem with subexponential sparsity already implies longstanding circuit lower bounds. In fact, we have an equivalence:

Theorem 1.7. *$\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$ if and only if there exists an $\varepsilon > 0$ such that for every sufficiently small $\beta > 0$, there is a $2^{n^{\beta}}$-sparse language $L \in \mathsf{NTIME}[2^{n^{\beta}}]$ without $n^{1+\varepsilon}$-size circuits.*

It follows that an $n^{1+\varepsilon}$-size circuit lower bound for $\mathsf{MCSP}[2^{m/\log^{\star} m}]$ implies $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$. We remark while the lower bound consequence here is *much weaker* than the consequences of prior work [OS18, OPS19, MMW19] (only $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$, instead of $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$), the hypothesis has much more flexibility: Theorem 1.7 allows for any sparse language in $\mathsf{NTIME}[2^{n^{o(1)}}]$, while the MCSP problem is in $\mathsf{NTIME}[n^{1+o(1)}]$.[7]

[7]We remark that these results are not directly related to hardness magnification for NC^1-complete problems [AK10, CT19], as the problems studied in these works are clearly not sparse.

Finally, we observe that Theorem 1.7 is similar in spirit to the Hartmanis-Immerman-Sewelson theorem [HIS85] which states that there is a polynomially-sparse language in $\mathsf{NP} \setminus \mathsf{P}$ if and only if $\mathsf{NE} \neq \mathsf{E}$. Theorem 1.7 can be interpreted as a certain optimized, non-uniform analogue of Hartmanis-Immerman-Sewelson theorem, in a different regime of sparsity.

Organization of the Paper. In Section 2, we introduce the necessary preliminaries for this paper. In Section 3, we prove that fixed-polynomial circuit lower bounds for P^{NP} is equivalent to a (weak) Karp-Lipton theorem for P. In Section 4, we prove our equivalence theorem for NP circuit lower bounds, fast simulations of NP, and nondeterministic polynomial-time derandomization, under the hypothesis $\mathsf{NP} \subset \mathsf{P}/\operatorname{poly}$. In Section 5, we show how our equivalence theorem implies that fixed polynomial circuit lower bounds for NP implies better Karp-Lipton theorems for higher complexity classes. In Section 6, we prove the consequences of weak circuit lower bounds for sparse NP languages. Finally, in Section 7, we discuss some interesting open questions stemming from this work.

2 Preliminaries

We assume basic knowledge of complexity theory (see e.g. [AB09, Gol08] for excellent references). Here we review some notation and concepts that are of particular interest for this paper.

Notation. All languages considered are over $\{0, 1\}$. For a language L, we define $L_n := \{0, 1\}^n \cap L$. For $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}[s(n)]$ is the class of languages decided by an infinite circuit family where the nth circuit in the family has size at most $s(n)$. $\oplus\mathsf{P}$ is the closure under polynomial-time reductions of the decision problem Parity-SAT: *Given a Boolean formula, is the number of its satisfying assignments odd?*

For a deterministic or nondeterministic class $\mathcal{C}$ and function $a(n)$, $\mathcal{C}/a(n)$ is the class of languages L such that there is an $L' \in \mathcal{C}$ and function $f : \mathbb{N} \to \{0, 1\}^\star$ with $|f(n)| \leq a(n)$ for all x, such that $L = \{x \mid (x, f(|x|)) \in L'\}$. That is, the advice string $f(n)$ can be used to solve all n-bit instances within class $\mathcal{C}$. For "promise" classes $\mathcal{C}$ such as MA and AM, $\mathcal{C}/a(n)$ is defined similarly, except that the promise of the class is only required to hold when the correct advice $f(n)$ is provided.

2.1 Infinitely Often and Robust Simulations

In this section, let $\mathscr{C}$ be a class of languages. Here we recall infinitely often and robust simulations, the latter of which was first defined and studied in [FS17]. Robust simulations expand on the notion of "infinitely often" simulations. A language $L \in$ i.o.-$\mathscr{C}$ (*infinitely often* $\mathscr{C}$), if there is a language L' in $\mathscr{C}$ such that there are infinitely many n such that $L_n = L'_n$. A language $L \in$ r.o.-$\mathscr{C}$ (*robustly often* $\mathscr{C}$), if there is a language L' in $\mathscr{C}$ such that for all $k \geq 1$, there are infinitely many n such that $L_m = L'_m$ for all $m \in [n, n^k]$. In this case, we say L' *r.o.-computes* L.

c-Robust Simulations. We consider a parameterized version of the robust simulation concept which is useful for stating our results. Let $c \geq 1$ be an integer constant. We say a language $L \in$ c-r.o.-$\mathscr{C}$ (c-robustly often $\mathscr{C}$) if there is an $L' \in \mathscr{C}$ and infinitely many n such that $L_m = L'_m$ for all $m \in [n, n^c]$. In this case, we say L' c-r.o.-computes L. Note that $L \in$ r.o.-$\mathscr{C}$ implies $L \in$ c-r.o.-$\mathscr{C}$ for all c, but the converse is not necessarily true.

More generally, a property $P(n)$ holds c-robustly often (c-r.o.-) if for all integers k, there are infinitely many m's such that $P(n)$ holds for all $n \in [m, m^c]$.

2.2 Non-deterministic Pseudo-Random Generators

Let $w(n), s(n) : \mathbb{N} \to \mathbb{N}$, and let $\mathscr{C}$ be a class of functions. We say a function family G, specified by $G_n : \{0,1\}^{w(n)} \times \{0,1\}^{s(n)} \to \{0,1\}^* \cap \{\perp\}$, is a *nondeterministic PRG against* $\mathscr{C}$ if for all sufficiently large n and all $C \in \mathscr{C}$, the following hold:

- For all $y \in \{0,1\}^{w(n)}$, either $G_n(y,z) \neq \perp$ for all z's (such a y is called *good*), or $G_n(y,z) = \perp$ for all z's (a *bad* y).
- There is at least one good $y \in \{0,1\}^{w(n)}$.
- Suppose $y \in \{0,1\}^{w(n)}$ is good, C has m input bits, and $|G_n(y,z)| \geq m$ for all z. Then

$$\left| \Pr_{z \in \{0,1\}^{s(n)}}[C(G_n(y,z)) = 1] - \Pr_{z \in \{0,1\}^m}[C(z) = 1] \right| < 1/n.$$

  As usual, if C takes less than $|G_n(y,z)|$ inputs, $C(G_n(y,z))$ corresponds to feeding C with the first m bits of $G_n(y,z)$.

Usually we are only interested in the seed length parameter $s(n)$ and the running time $T(n)$ of the PRG G_n as a function of n. To be concise, we say G is a $T(n)$*-time NPRG of seed length* $s(n)$ *against* $\mathscr{C}$.

We say G is a i.o.-NPRG or r.o.-NPRG, if it only fools functions in $\mathscr{C}$ infinite often or robustly often.

2.3 Circuit Complexity of Strings and Pseudorandom Generators

For a circuit C on ℓ inputs, we define the truth-table of C, denoted $tt(C) \in \{0,1\}^{2^\ell}$, to be the evaluation of C on all possible inputs sorted in lexicographical order. For every string y, let 2^ℓ be the smallest power of 2 such that $2^\ell > |y|$. We define the circuit complexity of y, denoted as $CC(y)$, to be the circuit complexity of the ℓ-input function defined by the truth-table $y10^{2^\ell - |y| - 1}$. We will use the following strong construction of pseudorandom generators from hard functions:

Theorem 2.1 (Umans [Uma03])**.** *There is a constant g and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for all s and Y satisfying $CC(Y) \geq s^g$, and for all circuits C of size s,*

$$\left| \Pr_{x \in \{0,1\}^{g \log |Y|}}[C(G(Y,x)) = 1] - \Pr_{x \in \{0,1\}^s}[C(x) = 1] \right| < 1/s.$$

Furthermore, G is computable in $\mathrm{poly}(|Y|)$ *time.*

Fortnow-Santhanam-Williams [FSW09]. A work related to this paper is that of Fortnow, Santhanam, and Williams, who proved the equivalences $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all $k \iff \mathsf{P}^{\mathsf{NP}[n^k]} \not\subset \mathsf{SIZE}[n^c]$ for all k, c and $\mathsf{AM} \not\subset \mathsf{SIZE}[n^k]$ for all $k \iff \mathsf{MA} \not\subset \mathsf{SIZE}[n^k]$ for all k. We use intermediate results of theirs in our equivalence theorems (see the citations).

3 P^{NP} Circuit Lower Bounds Equivalent to Karp-Lipton Collapses to P^{NP}

In this section we prove Theorem 1.1 (restated below).

Reminder of Theorem 1.1. *$\mathsf{P}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[n^k]$ for all k if and only if ($\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{PH} \subset$ i.o.-$\mathsf{P}^{\mathsf{NP}}_{/n}$).*

We begin with a lemma on the simulation of poly-time functions with an NP oracle. Essentially it says that if functions with an NP oracle always output strings of low circuit complexity, then we can simulate P^{NP} extremely efficiently in the polynomial hierarchy. This is similar in spirit to Fortnow, Santhanam, and Williams' result that $\mathsf{P}^{\mathsf{NP}} \subset \mathsf{SIZE}[n^k]$ implies $\mathsf{NP} \subseteq \mathsf{MATIME}(n^{O(k)})$ [FSW09]; our result is more complex in that we simulate all of P^{NP}.

Lemma 3.1. *Suppose there is a k such that for all $\mathsf{FP}^{\mathsf{NP}}$ functions f, the circuit complexity of $f(x)$ is at most $|x|^k$ for all but finite many x. Then $\mathsf{P}^{\mathsf{NP}} \subseteq \Sigma_3\mathsf{TIME}[n^{O(k)}]$.*

Proof. Let $L \in \mathsf{P}^{\mathsf{NP}}$ be a language which can be computed by a $\mathsf{3\text{-}SAT}$ oracle machine M in n^c time, for a constant c. Without loss of generality, we may assume M is a single-tape machine.

The $\mathsf{FP}^{\mathsf{NP}}$ Function f_{sol}. Consider the following $\mathsf{FP}^{\mathsf{NP}}$ function f_{sol}:

$\mathsf{FP}^{\mathsf{NP}}$ function f_{sol} for printing assignments to all satisfiable oracle queries

- Given an input x, simulate the $\mathsf{3\text{-}SAT}$ oracle machine M running on the input x.
- On the i-th step, if M makes an oracle query ψ (ψ is a $\mathsf{3\text{-}SAT}$ instance) and ψ is satisfiable, call the NP oracle multiple times to construct a satisfying assignment for ψ, and print it. Letting m be the length of the assignment (note that $m \le n^c$), we print $n^c + 1 - m$ additional ones.
- Otherwise, print $n^c + 1$ zeros on the i-th step.

In the following we always assume n is sufficiently large. For all x with $|x| = n$, by assumption we know the string $f_{\mathsf{sol}}(x)$ has an n^k size circuit. Let ψ be a $\mathsf{3\text{-}SAT}$ query made on i-th step which is satisfiable; ψ has a satisfying assignment corresponding to a sub-string of $f_{\mathsf{sol}}(x)$ starting from the position $(i-1)\cdot(n^c+1)+1$, and therefore has circuit complexity at most $O(n^k) \le n^{k+1}$. In particular, we can define a circuit $E_i(j) := f_{\mathsf{sol}}(x)((i-1)\cdot(n^c+1)+j)$ whose truth table encodes a SAT assignment to ψ.

The $\mathsf{FP}^{\mathsf{NP}}$ Function f_{history}. Next, we define a function $\mathsf{FP}^{\mathsf{NP}}$ function f_{history}, which prints the computation history of M. More precisely, we can interpret $f_{\mathsf{history}}(x)$ as a matrix $\mathsf{cell}(x) \in \Sigma^{n^c \times n^c}$, such that $\mathsf{cell}(i,j)$ represents the state of the j-th cell of the working tape before the i-th step, and Σ is a constant-size alphabet which represents all possible states of a cell. From our assumption, for an x with $|x| = n$, we know that $f_{\mathsf{history}}(x)$ has an n^k-size circuit.

The Algorithm. Now we are ready to describe a Σ_3 algorithm for L running in $n^{O(k)}$ time. At a high level, the algorithm first guesses two circuits C_{history} and C_{sol}, whose truth-tables are supposed to represent $f_{\mathsf{history}}(x)$ and $f_{\mathsf{sol}}(x)$, it tries to verify that these circuits correspond to a correct accepting computation of M on x. The whole verification can be done in $\Pi_2\mathsf{TIME}[n^{O(k)}]$, utilizing the fact that M is making $\mathsf{3\text{-}SAT}$ queries. The formal description of the algorithm is given below.

A $\Sigma_3\mathsf{TIME}[n^{O(k)}]$ algorithm for L

(1) Given an input x, guess two n^k-size circuits C_{history} and C_{sol} where the truth-table of C_{history} is intended to be $f_{\mathsf{history}}(x)$), and the truth-table of C_{sol} is intended to be $f_{\mathsf{sol}}(x)$. Let $\mathsf{cell} \in \Sigma^{n^c \times n^c}$ be the matrix (tableau) corresponding to the truth-table of C_{history}.

(2) We check that C_{history} is consistent and accepting, assuming its claimed answers to oracle queries are correct. In particular, we universally check over all $(i,j) \in [n^c] \times [n^c]$ that $\mathsf{cell}(i,j)$ is consistent with the contents of $\mathsf{cell}(i-1,j-1), \mathsf{cell}(i-1,j), \mathsf{cell}(i,j+1)$ when $i > 1$, whether it agrees with the initial configuration when $i = 1$, and whether M is in an accept state when $i = n^c$.

(3) We check that the claimed answers to oracle queries in C_{history} are correct. For convenience, we assume the query string always starts at the leftmost position on the tape. We universally check over all step $i \in [n^c]$:

If there is no query at the i-th step, we *accept*.

(A) Let ψ be the $\mathsf{3\text{-}SAT}$ query. If the claimed answer in C_{history} for ψ is *yes*, we examine the corresponding sub-string of $tt(C_{\mathsf{sol}})$, and check universally over all clauses in ψ that it is satisfied by the corresponding assignment in $tt(C_{\mathsf{sol}})$ (accepting if the check passes and rejecting if it fails).

(B) If the claimed answer in C_{history} for ψ is *no*, we universally check over all n^{k+1}-size circuits D that $tt(D)$ is not an assignment to ψ, by existentially checking that there is a clause in ψ which is not satisfied by $tt(D)$.

Running Time. It is straightforward to see that the above is a $\Sigma_3\mathsf{TIME}[n^{O(k)}]$ algorithm.

Correctness. When $x \in L$, there are C_{sol} and C_{history} such that $tt(C_{\mathsf{sol}})$ and $(\mathsf{C}_{\mathsf{history}})$ correspond to $f_{\mathsf{sol}}(x)$ and $f_{\mathsf{history}}(x)$, so all of the checks pass and the above algorithm accepts x.

Let $x \notin L$. We want to show that all possible n^k-size circuits for C_{history} and C_{sol} will be rejected. Assume for contradiction that there are circuits C_{history} and C_{sol} that can pass the whole verification. By our checks in step (2) of the algorithm, C_{history} is consistent and ends in accept state; therefore, at least one answer to its oracle queries is not correct. Suppose the first incorrect answer occurs on the i-th step. Since C_{history} is consistent and all queries made before the i-th are correctly answered, the i-th query ψ is actually the correct i-th query made by machine M on the input x.

Therefore, if the correct answer to ψ is yes but C_{history} claims it is no, case (B) will not be passed, as there is always a satisfying assignment that can be represented by the truth-table of an n^{k+1}-size circuit. Similarly, if C_{history} incorrectly claims the answer is yes, then case (A) cannot be passed, as ψ is unsatisfiable. □

We are now ready to prove Theorem 1.1.

Proof of Theorem 1.1. Suppose (1) P^{NP} does not have $\mathsf{SIZE}[n^k]$ circuits for any fixed k and (2) $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$. By assumption (2), we have that for every c, $\Sigma_3\mathsf{TIME}[n^c] \subset \mathsf{SIZE}[n^{O(c)}]$. Therefore, applying (1), $\mathsf{P}^{\mathsf{NP}} \not\subseteq \Sigma_3\mathsf{TIME}[n^c]$ for every c. By the contrapositive of Lemma 3.1, for every k there is a P^{NP} function B that for infinitely many x of length n, the circuit complexity of $B(x)$ is greater than n^k. In other words, $B(x)$ outputs the truth tables of hard functions on infinitely many x.

Assumption (2) also implies a collapse of the polynomial hierarchy to $\mathsf{ZPP}^{\mathsf{NP}}$ [KW98]. By (2), we also have $\mathsf{ZPP}^{\mathsf{NP}} \subset \mathsf{P}_{/\mathsf{poly}}$, so every $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm A has polynomial-size circuits, and thus by standard hardness-to-PRG constructions (e.g., Theorem 2.1) there is a fixed k such that a string of circuit complexity at least n^k can be used to construct a PRG that fools algorithm A on inputs of length n. As shown above, there is a function B in P^{NP} that can produce such strings on infinitely many inputs x. If the inputs x that make B produce high complexity strings are given as advice, then the $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm A can be simulated in $\mathsf{P}^{\mathsf{NP}}_{/n}$: first, call B on the advice x to generate a hard function, produce a PRG of seed length $O(\log n)$ with the hard function, then simulate A on the input and the pseudorandom strings output by the PRG, using the NP oracle to simulate the NP oracle of A. Thus we have $\mathsf{ZPP}^{\mathsf{NP}} \subset$ i.o.-$\mathsf{P}^{\mathsf{NP}}/n$.

Finally, we note that the n bits of advice can be reduced to n^{ε} bits for any desired $\varepsilon > 0$. For every $k > 0$, we can find an $\mathsf{FP}^{\mathsf{NP}}$ function that outputs a string of circuit complexity greater than n^k. Setting $k' = k/\varepsilon$, we can use an n^{ε}-length input as advice, and still get a function that is hard enough to derandomize $((n^{\varepsilon})^{k'} = (n^{\varepsilon})^{k/\varepsilon} = n^k)$. □

4 An Equivalence Theorem Under $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$

In this section we prove Theorem 1.2 together with several applications.

First, we need a strong size lower bound for a language in $(\mathsf{MA} \cap \mathsf{coMA})/1$. The proof is based on a similar lemma in a recent work [Che19] (which further builds on [MW18, San09]). We present a proof in Appendix A for completeness.

Lemma 4.1 (Implicit in [Che19])**.** *For all constants k, there is an integer c, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})_{/1}$, such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^{\tau}$, either*

- *$\mathsf{SIZE}(L_n) > n^k$, or*
- *$\mathsf{SIZE}(L_m) > m^k$, for an $m \in (n^c, 2 \cdot n^c) \cap \mathbb{N}$.*

We also need the following two simple lemmas.

Lemma 4.2. *NP is not in $\mathsf{SIZE}[n^k]$ for all k iff $\mathsf{NP}_{/n}$ is not in $\mathsf{SIZE}[n^k]$ for all k.*

Proof. The $\Rightarrow$ direction is trivial. For the $\Leftarrow$ direction, suppose NP is in $\mathsf{SIZE}[n^k]$ for an integer k. Let $L \in \mathsf{NP}_{/n}$, and M and $\{\alpha_n\}_{n\in\mathbb{N}}$ be its corresponding nondeterministic Turing machine and advice sequence. Let $p(n)$ be a polynomial running time upper bound of M on inputs of length n.

Now, we define a language L' such that a pair $(x, \alpha) \in L'$ if and only if $|x| = |\alpha|$ and M accepts x with advice bits set to α in $p(|x|)$ steps. Clearly, $L' \in \mathsf{NP}$ from the definition, so it has an n^k-size circuit family. Fixing the advice bits to the actual α_n's in the circuit family, we have an $n^{O(k)}$-size circuit family for L as well. This completes the proof. □

Lemma 4.3 (Theorem 14 [FSW09])**.** *Let k be an integer. If $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$ and all NP verifiers have n^k-size witnesses, then $\mathsf{NP} \subseteq \mathsf{MATIME}[n^{O(k)}] \subset \mathsf{SIZE}[n^{O(k)}]$.*

Proof. Assume all NP verifiers have n^k-size witnesses. By guessing circuits for the witnesses to PCP verifiers, it follows that $\mathsf{NP} \subseteq \mathsf{MATIME}[n^{O(k)}]$ [FSW09]. Furthermore, we have $\mathsf{MATIME}[n^{O(k)}] \subset \mathsf{NTIME}[n^{O(k)}]_{/n^{O(k)}} \subset \mathsf{SIZE}[n^{O(k)}]$. The last step follows from the assumption that $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$ (and therefore $\mathsf{SAT} \in \mathsf{SIZE}[n^c]$ for a constant c). □

Now, we are ready to prove our equivalence theorem (restated below).

Reminder of Theorem 1.2. *Assuming $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$, the following are equivalent:*

1. *NP is not in $\mathsf{SIZE}[n^k]$ for all k.*
2. *$\mathsf{AM}_{/1}$ is in c-r.o.-$\mathsf{NP}_{/n^{\varepsilon}}$ for all $\varepsilon > 0$ and integers c.*
3. *NP does not have n^k-size witnesses for all k.*[8]

[8]See the statement of Theorem 1.2 in the introduction for the definition of n^k-size witnesses.

4. *For all k and d, there is a poly-time nondeterministic PRG with n bits of advice against n^k-size circuits d-robustly often.*[9]

5. *NP is not in $\mathsf{AMTIME}(n^k)$ for all k.*

6. *$(\mathsf{NP} \cap \mathsf{coNP})_{/n^\varepsilon}$ is not in $\mathsf{SIZE}[n^k]$ for all k and all $\varepsilon > 0$.*

7. *$(\mathsf{AM} \cap \mathsf{coAM})_{/1}$ is in c-r.o.-$(\mathsf{NP} \cap \mathsf{coNP})_{/n^\varepsilon}$ for all $\varepsilon > 0$ and all integers c.*

Proof. We prove the following directions to show equivalence.

(2) $\Rightarrow$ (1). Suppose (2) holds. For all k, let L and c be the $\mathsf{MA}_{/1}$ language and the corresponding constant c guaranteed by Lemma 4.1. By (2) and the fact that $\mathsf{MA}_{/1} \subseteq \mathsf{AM}_{/1}$, there is an $\mathsf{NP}_{/n}$ language L' such that for infinitely many n's, L' agrees with L on inputs with length in $[n, n^{2c}]$.

Let $\tau = \lceil \log(n) \rceil$. By the condition of Lemma 4.1, we know that for at least one $\ell \in [n, n^{2c}]$, we have $\mathsf{SIZE}(L'_\ell) \geq \ell^k$. Since there are infinitely many such n, we conclude that L' is not in $\mathsf{SIZE}[n^k]$. Since k can be an arbitrary integer, it further implies that $\mathsf{NP}_{/n}$ is not in $\mathsf{SIZE}[n^k]$ for all k, and hence also NP is not in $\mathsf{SIZE}[n^k]$ for all k by Lemma 4.2.

(1) $\Rightarrow$ (3). We prove the contrapositive. Suppose NP has n^k-size witnesses for an integer k. Then, by Lemma 4.3, $\mathsf{NP} \subset \mathsf{SIZE}[n^{O(k)}]$.

(3) $\Rightarrow$ (4). This more-or-less follows directly from standard hardness-to-pseudorandomness constructions [Uma03]. More specifically, for all integers k and d and $\varepsilon > 0$, there is a language $L \in \mathsf{NP}$ without $n^{gkd/\varepsilon}$-size witnesses. Equivalently, there is a poly-time verifier V for L, such that there are infinitely many $x \in L$ such that for all y with $V(x, y) = 1$, it follows $CC(y) \geq |x|^{gkd/\varepsilon}$.

For such an $x \in L$ with $|x| = m$, we can guess a y such that $V(x, y) = 1$ and apply Theorem 2.1 to construct a poly-time nondeterministic PRG with seed length $O(\log m)$, which works for input length $n \in [m^{1/\varepsilon}, m^{d/\varepsilon}]$ and against n^k-size circuits. Note that advice length is $|x| = m \leq n^\varepsilon$.

(4) $\Rightarrow$ (2). First, under the assumption that $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$, we have the collapse $\mathsf{AM}_{/1} = \mathsf{MA}_{/1}$ [AKSS95]. So it suffices to show that $\mathsf{MA}_{/1} \subset$ c-r.o.-$\mathsf{NP}_{/n^\varepsilon}$ for all $\varepsilon > 0$ and integers d.

Let $L \in \mathsf{MA}_{/1}$. That is, for a constant k, there is an n^k-time algorithm $A(x, y, z, \alpha)$ with one bit of advice α_n, such that

- $x \in L \Rightarrow$ there is a y of $|x|^k$ length such that $\Pr_z[A(x, y, z, \alpha_n) = 1] \geq 2/3$.
- $x \notin L \Rightarrow$ for all y of $|x|^k$ length, $\Pr_z[A(x, y, z, \alpha_n) = 1] \leq 1/3$.

Fixing the x, y, α_n, we can construct a circuit $C_{x,y,\alpha_n}(z) := A(x, y, z, \alpha_n)$ of size n^{2k} in n^{2k} time.

Now, by (4), for all d, there is a poly-time NPRG G with seed length $O(\log n)$ and advice length n^ε such that there are infinitely many m's such that for all $n \in [m, m^d]$, G_n fools n^{2k}-size circuits.

Applying G_n to fool C_{x,y,α_n} directly, we have a language $L' \in \mathsf{NP}_{/n^\varepsilon}$ such that there are infinitely many m such that L' agrees with L on all input lengths in $[m, m^d]$. This completes the proof since d can be made arbitrarily large.

(5) $\Rightarrow$ (3). We prove the contrapositive. Suppose NP has n^k-size witnesses for an integer k. By Lemma 4.3, it follows that $\mathsf{NP} \subseteq \mathsf{MATIME}[n^{O(k)}] \subseteq \mathsf{AMTIME}[n^{O(k)}]$.

(1) $\Rightarrow$ (5). Again, we prove the contrapositive. We have $\mathsf{NP} \subseteq \mathsf{AMTIME}[n^{O(k)}] \subset \mathsf{NTIME}[n^{O(k)}]_{/n^{O(k)}} \subset \mathsf{SIZE}[n^{O(k)}]$. The last step follows from the assumption that $\mathsf{NP} \subseteq \mathsf{P}_{/\mathsf{poly}}$ (and therefore $\mathsf{SAT} \in \mathsf{SIZE}[n^c]$ for a constant c).

[9] See the Preliminaries for a full definition of nondeterministic PRG and d-robustly often.

(6) ⇒ (1). $(\mathsf{NP} \cap \mathsf{coNP})_{/n^\varepsilon}$ is not in $\mathsf{SIZE}[n^k]$ for all k and $\varepsilon > 0$ implies $\mathsf{NP}_{/n}$ is not in $\mathsf{SIZE}[n^k]$ for all k, which in turn implies NP is not in $\mathsf{SIZE}[n^k]$ for all k by Lemma 4.2.

(4) ⇒ (7). This follows similarly as the direction from (4) to (2).

(7) ⇒ (6). This follows similarly as the direction from (2) to (1). Note that [AKSS95] also implies $(\mathsf{MA} \cap \mathsf{coMA})_{/1} = (\mathsf{AM} \cap \mathsf{coAM})_{/1}$ under the assumption $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}}$. □

5 NP Circuit Lower Bounds Imply Better Karp-Lipton Collapses

Now we show a corollary of Theorem 1.2 that NP circuit lower bounds imply better Karp-Lipton collapses.

Reminder of Theorem 1.4. *Let $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PSPACE}, \mathsf{PP}, \mathsf{EXP}\}$. Suppose $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k. Then for all $\varepsilon > 0$, ($\mathcal{C} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathcal{C} \subset$ i.o.-$\mathsf{NP}_{/n^\varepsilon}$). In particular, polynomial-size circuits for any $\mathcal{C}$-complete language can be constructed in NP on infinitely many input lengths with n^ε advice.*

Proof of Theorem 1.4. We first prove it for $\oplus\mathsf{P}$. Suppose for all k, $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ and $\oplus\mathsf{P} \subset \mathsf{P}_{/\mathsf{poly}}$.

First, note that $\mathsf{BPP}^{\oplus\mathsf{P}} \subset \mathsf{P}_{/\mathsf{poly}}$, implying $\mathsf{PH} \subset \mathsf{P}_{/\mathsf{poly}}$ by Toda's theorem [Tod91]. Therefore, by Theorem 1.2 together with our assumption, we have $\mathsf{MA} \subset$ c-r.o.-$\mathsf{NP}_{/n^\varepsilon}$ for all $\varepsilon > 0$ and integers c. In particular, $\mathsf{MA} \subset$ i.o.-$\mathsf{NP}_{/n^\varepsilon}$ for all $\varepsilon > 0$. Now it suffices to show that $\oplus\mathsf{P} \subset \mathsf{P}_{/\mathsf{poly}} \implies \oplus\mathsf{P} \subseteq \mathsf{MA}$.

Let Π be the random self-reducible and downward self-reducible $\oplus\mathsf{P}$-complete language in [IKV18]. By our assumption that $\oplus\mathsf{P} \subset \mathsf{P}_{/\mathsf{poly}}$, Π has a poly-size circuit family.

Then we can guess-and-verify these circuits in MA. We first existentially guess a circuit C_k for Π on every input length $k = 1, \ldots, n$. C_1 can be verified in constant time, and each successive circuit can be verified via random downward self-reducibility: given a circuit of length m that computes Π_m exactly, a circuit of length $m + 1$ can be checked on random inputs to verify (with high probability) its consistency with Π_{m+1} (which is computable using the downward self-reducibility and the circuit for Π_m). Then we can apply the random self-reducibility to construct an exact circuit for Π_{m+1} from C_{m+1} with high probability, as we already know C_{m+1} approximates Π_{m+1} very well. Therefore, with high probability, we can guess-and-verify a circuit for Π_n via a poly-time MA computation. This puts $\oplus\mathsf{P} \subseteq \mathsf{MA}$. Combining that with $\mathsf{MA} \subset$ i.o.-$\mathsf{NP}_{/n^\varepsilon}$ for all $\varepsilon > 0$, we can conclude that $\oplus\mathsf{P} \subset$ i.o.-$\mathsf{NP}_{/n^\varepsilon}$ for all $\varepsilon > 0$.

To construct a circuit for Π_n in i.o.-$\mathsf{NP}_{/n^\varepsilon}$, note that by Theorem 1.6, for all k, we have an i.o.-NPRG fooling n^k-size circuits. We can pick k to be a sufficiently large integer, and use the i.o.-NPRG to derandomize the above process. This turns out to be more subtle than one might expect.

Construction of poly-size circuits of Π_n in i.o.-$\mathsf{NP}_{/n^\varepsilon}$. Let d be a sufficiently large constant. Since we only aim for an i.o.-construction, we can assume that our i.o.-NPRG works for the parameter n, and fools all n^d-size circuits. Also, suppose we have $\mathsf{SIZE}(\Pi_n) \le n^c$ for all n and a constant c.

We say a circuit C γ-approximates a function f, if $C(x) = f(x)$ for at least a γ fraction of the inputs.

Again, suppose we already constructed the circuits $C_1, C_2, \ldots, C_k$ for $\Pi_1, \Pi_2, \ldots, \Pi_k$. This time we cannot guarantee C_i exactly computes Π_i. Instead, we relax the condition a bit and ensure that C_i $(1-4/n)$-approximates Π_i for all $i \in [k]$. Clearly, we can check $C_1 \equiv \Pi_1$ directly so this can be satisfied when $k = 1$.

We now show how to construct an approximate circuit for Π_{k+1}. First, using the random self-reducibility of Π and the circuit C_k approximating Π_k, there is an oracle circuit E of size $\mathrm{poly}(n)$, which takes two inputs x with $|x| = k$ and r with $|r| = \mathrm{poly}(n)$, such that for all x,

$$\Pr_r \left[E^{C_k}(x, r) = \Pi_k(x)\right] \ge 1 - 1/2^n.$$

Also, by the downward self-reducibility of Π, there is an oracle machine D of $\mathrm{poly}(k)$ size, such that $D^{\Pi_k}(z) = \Pi_{k+1}(z)$ for all z.

Now, consider the following circuit $G(x, r)$ for computing Π_{k+1}: the circuit simulates D^{Π_k}, while answering all queries w to Π_k using $E^{C_k}(w, r)$. For each input $x \in \{0, 1\}^{k+1}$, let $w_1, w_2, \ldots, w_{\mathrm{poly}(n)}$ be all queries to Π_k made by running D on the input x assuming all answers are correct, we can see that if $E^{C_k}(w_j, r) = \Pi_k(w_j)$ for all these w_j's, then $G(x, r) = \Pi_{k+1}(x)$. Therefore, we have

$$\Pr_r[G(x, r) = \Pi_{k+1}(x)] \geq 1 - \mathrm{poly}(n)/2^n,$$

for all $x \in \{0, 1\}^{k+1}$.

Now, we guess a circuit C_{k+1} of size $(k + 1)^c$ which is supposed to compute Π_{k+1}. By an enumeration of all possible seeds to our NPRG, we can estimate the probability

$$p_{\mathsf{good}} := \Pr_{x \in \{0,1\}^{k+1}} \Pr_r[G(x, r) = C_{k+1}(x)].$$

within $1/n$ in $\mathrm{poly}(n)$ time, as the expression $[G(x, r) = C_{k+1}(x)]$ has a $\mathrm{poly}(n)$ size circuit with inputs being x and r. Let our estimation be p_{est}. We have $|p_{\mathsf{good}} - p_{\mathsf{est}}| \leq 1/n$.

Putting the above together, we have

$$\left| \Pr_{x \in \{0,1\}^{k+1}} [\Pi_{k+1}(x) = C_{k+1}(x)] - p_{\mathsf{good}} \right| \leq \mathrm{poly}(n)/2^n.$$

We reject immediately if our estimation $p_{\mathsf{est}} < 1 - 2/n$ (note that if C_{k+1} is the correct circuit, p_{good} would be larger than $1 - \mathrm{poly}(n)/2^n > 1 - 1/n$, and therefore $p_{\mathsf{est}} > 1 - 2/n$). So after that, we can safely assume that C_{k+1} $(1 - 4/n)$-approximates Π_{k+1}.

Therefore, at the end we have an n^c-size circuit C_n which $(1 - 4/n)$-approximates Π_n, and we try to recover an exact circuit for Π_n from C_n by exploiting the random self-reducibility of Π_n again. Note that there is an oracle circuit $E(x, r)$, which takes two inputs x with $|x| = n$ and r with $|r| = \mathrm{poly}(n)$ such that for all x,

$$\Pr_r[E^{C_n}(x, r) = \Pi_n(x)] \geq 2/3.$$

Now, we generate $\ell = n^{O(1)}$ strings $r_1, r_2, \ldots, r_\ell$ by enumerating all seeds to the NPRG. We construct our final circuit C to be the majority of $E^{C_n}(x, r_j)$ for all $j \in [\ell]$. It is not hard to see that C computes Π_n exactly, as our inputs $\{r_j\}_{j \in [\ell]}$ fool the expression $\left[E^{C_n}(x, r) = \Pi_n(x)\right]$ for all $x \in \{0, 1\}^n$.

For the case of PP and PSPACE, one can implement the above procedure in the same way, using the corresponding random self-reducible and downward self-reducible PP-complete and PSPACE-complete languages (Permanent and the PSPACE-complete language in [TV07]).

For the case of EXP, note that $\mathsf{EXP} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{EXP} = \mathsf{PSPACE}$, so we can proceed the same way as for PSPACE (since $\mathsf{EXP} = \mathsf{PSPACE}$, PSPACE-complete languages are also EXP-complete). □

6 Consequence of Weak Circuit Lower Bounds for Sparse Languages in NP

Now, we are ready to prove the consequences of weak circuit lower bounds for sparse NP languages. We first need the following lemma.

Lemma 6.1 (Hardness Verification from Circuit Lower Bounds for Sparse $\mathsf{NTIME}[T(n)]$ Languages)**.** *Let $S_{\mathsf{ckt}}(n), S_{\mathsf{sparse}}(n), T(n) : \mathbb{N} \to \mathbb{N}$ be time constructible functions. Suppose there is an $S_{\mathsf{sparse}}(n)$-sparse language $L \in \mathsf{NTIME}[T(n)]$ without $(n \cdot S_{\mathsf{ckt}}(n))$-size circuits. Then there is a procedure V such that:*

- *V takes a string z of length $n \cdot S_{\mathsf{sparse}}(n)$ as input and an integer $\ell \leq S_{\mathsf{sparse}}(n)$ as advice.*

- *V runs in $O(S_{\mathsf{sparse}}(n) \cdot T(n))$ nondeterministic time.*

- *For infinitely many n, there is an integer $\ell_n \leq S_{\mathsf{sparse}}(n)$ such that $V(z, \ell_n)$ accepts exactly one string z, and z has circuit complexity $\Omega(S_{\mathsf{ckt}}(n)/\log S_{\mathsf{sparse}}(n))$.*

Proof. Let L be the $\mathsf{NTIME}[T(n)]$ language in the assumption. Let $N = n \cdot S_{\mathsf{sparse}}(n)$. We define a string $\mathsf{List}_{L_n} \in \{0,1\}^N$ as the concatenation of all $x \in L_n$ in lexicographical order, together with additional zeros at the end to make the string have length exactly N.

Now define a function f_n on $m = \log\lceil N+1 \rceil$ bits, with truth-table $\mathsf{List}_{L_n}10^{2^m - N}$.

We claim that $\mathsf{SIZE}(L_n) \leq O(\mathsf{SIZE}(f_n) \cdot n \cdot \log(S_{\mathsf{sparse}}(n)))$. To determine whether $x \in L_n$, it would suffice to perform a binary search on the list List_{L_n}. We construct a circuit for L_n which performs binary search using f_n. First, we hard-wire the length of the list $\ell := |L_n| \leq S_{\mathsf{sparse}}(n)$ into our circuit for L_n so that the binary search can begin with the correct range. A binary search on $\mathsf{List}(L_n)$ takes $O(\log S_{\mathsf{sparse}}(n))$ comparisons, and each comparison requires $O(n)$ calls to f_n (to print the appropriate string). It is easy to see that the circuit size required for the binary search is dominated by the total cost of the comparisons; this proves the claim.

From the assumption, we know that for infinitely many n, L_n has no circuit of size $n \cdot S_{\mathsf{ckt}}(n)$. By our upper bound on the circuit size of L_n, it follows that on the same set of n, the function f_n has circuit complexity at least $\Omega(S_{\mathsf{ckt}}(n)/\log S_{\mathsf{sparse}}(n))$.

Now, we construct an algorithm V that only accepts the string $f_n = \mathsf{List}_{L_n}10^{2^m - N}$. We first need the integer $\ell = |L_n|$ as the advice. Given a string Y of length N, we check that Y contains exactly ℓ distinct inputs in $\{0,1\}^n$ in lexicographical order with the correct format, and we guess an $O(T(n))$-length witness for each input to verify they are indeed all in L. It is easy to see that V runs in $O(S_{\mathsf{sparse}}(n) \cdot T(n))$ nondeterministic time, which completes the proof. $\square$

Remark 6.2. *Note that the advice integer ℓ can be calculated directly with an NP oracle by doing a binary search for ℓ, which takes $O(\log S_{\mathsf{sparse}}(n))$ NP-oracle calls. That is, one can also use a $\mathsf{P}^{\mathsf{NP}[O(\log S_{\mathsf{sparse}}(n))]}$ verifier without advice bits in the statement of Lemma 6.1.*

Remark 6.3. *As mentioned in the introduction, the above proof can be seen as a type of* hardness condensation *for all sparse $\mathsf{NTIME}[T(n)]$ languages. The goal of hardness condensation [BS06, IJKW10] is that, given a hard function f on n input bits with complexity S, we want to construct a function $\widetilde{f}$ on $\ell \ll n$ input bits that still has complexity roughly S. The above proof shows any hard sparse language in $\mathsf{NTIME}[T(n)]$ can be "condensed" into a function representing its sorted yes-instances.*

Combing Lemma 6.1 with Theorem 2.1, we obtain a construction of an i.o.-NPRG.

Corollary 6.4 (NPRG from lower bounds against sparse $\mathsf{NTIME}[T(n)]$ languages)**.** *Under the circuit lower bound assumption of Lemma 6.1, there is an i.o.-NPRG G with the properties:*

- *G has $O(\log S_{\mathsf{sparse}}(n) + \log n)$ seed length.*

- *G takes $O(\log S_{\mathsf{sparse}}(n))$ bits of advice.*

- *G runs in $S_{\mathsf{sparse}}(n) \cdot T(n) + \mathrm{poly}(n \cdot S_{\mathsf{sparse}}(n))$ time.*

- *G fools circuits of size at most $(S_{\mathsf{ckt}}(n)/\log S_{\mathsf{sparse}}(n))^{\Omega(1)}$.*

Now we are ready to prove Theorem 1.6.

Reminder of Theorem 1.6. *Suppose there is an $\varepsilon > 0$, a $c \geq 1$, and an n^c-sparse $L \in \mathsf{NP}$ without $n^{1+\varepsilon}$-size circuits. Then $\mathsf{MA} \subset$ i.o.-$\mathsf{NP}_{/O(\log n)}$, $\mathsf{MA} \subseteq$ i.o.-$\mathsf{P}^{\mathsf{NP}[O(\log n)]}$, and $\mathsf{NE} \not\subset \mathsf{SIZE}[2^{\delta \cdot n}]$ for some $\delta > 0$ (which implies $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k).*

Proof. First, by Corollary 6.4 and setting $S_{\mathsf{ckt}}(n) = n^{\varepsilon}$, $S_{\mathsf{sparse}}(n) = n^c$ and $T(n) = \mathrm{poly}(n)$, there is an i.o.-NPRG with seed length $O(\log n)$ which takes $O(\log n)$ bits of advice, runs in $\mathrm{poly}(n)$ time, and fools circuits of size $n^{\Omega(\varepsilon)} = n^{\Omega(1)}$. Note that we can simply scale it up to make it fool circuits of size n^k for any k, with only a constant factor blowup on seed length and advice bits and a polynomial blowup on the running time.

Applying the i.o.-NPRG to arbitrary Merlin-Arthur computations, we conclude $\mathsf{MA} \subset$ i.o.-$\mathsf{NP}_{/O(\log n)}$. Similarly, $\mathsf{MA} \subseteq$ i.o.-$\mathsf{P}^{\mathsf{NP}[O(\log n)]}$ follows from Remark 6.2.

Now we show $\mathsf{NE} \not\subset \mathsf{SIZE}[2^{\delta \cdot n}]$ for some $\delta > 0$. By Lemma 6.1, there is a nondeterministic algorithm running in $\mathrm{poly}(n)$ time, given $\alpha_n = c \log n$ bits of advice, guess and verify a string of length n^{c+1} which has circuit complexity at least $n^{\varepsilon/2}$, for infinitely many n. We say these infinitely many n are *good* n.

Next, we define the following language $L \in \mathsf{NE}$: Given an input of length m. It treats the first $\ell = m/4c$ bits a binary encoded integer $n \leq 2^{\ell}$. Then it treats the next $c \log n$ input bits a as the advice, and tries to guess-and-verify a string z which passes the verification procedure in Lemma 6.1 with advice a and parameter n, and then it treats the next $(c+1) \cdot \log n$ input bits as an integer $i \in [n^{c+1}]$, and accepts if and only $z_i = 1$.

First, it is easy to verify $L \in \mathsf{NE}$, as the algorithm runs in $\mathrm{poly}(n) = 2^{O(\ell)} = 2^{O(m)}$ nondeterministic time. For the circuit complexity of L, we know that for the good n, on inputs of length of $m = 4 \cdot c \cdot \lceil \log n \rceil$, if we fix the first $m/4c$ bits to represent the integer n, and next $c \log n$ bit to the actual advice α_n, L would compute the hard string of length n^{c+1} on the next $(c+1) \cdot \log n$ bits. Therefore, $\mathsf{SIZE}(L_m) \geq n^{\varepsilon} \geq 2^{\Omega(m)}$ for infinitely many m's, which completes the proof. ☐

Finally, we prove Theorem 1.7.

Reminder of Theorem 1.7. *$\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$ if and only if there is an $\varepsilon > 0$ such that for all sufficiently small $\beta > 0$, there is a $2^{n^{\beta}}$-sparse language $L \in \mathsf{NTIME}[2^{n^{\beta}}]$ without $n^{1+\varepsilon}$-size circuits.*

Proof. **(⇒)** This direction is easy to prove using standard methods. Suppose $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$; this also implies $\mathsf{NE} \not\subset \mathsf{P}_{/\mathsf{poly}}$. Therefore, there is a language $L \in \mathsf{NTIME}[2^n]$ that does not have $n^{2/\beta}$-size circuits. Define a padded language $L' = \{x10^{|x|^{1/\beta}-1} | x \in L\}$. It is easy to see that $L' \in \mathsf{NTIME}[2^{m^{\beta}}]$, by running the NE algorithm for L on its first $n = O(m^{\beta})$ input bits. From the circuit lower bound on L, it follows that L' does not have $n^{2/\beta} = m^2$-size circuits.

(⇐) First, by Impagliazzo-Kabanets-Wigderson [IKW02], if for every ε and integer k, there is an i.o.-NPRG with seed length n^{ε}, n^{ε} advice bits, and $2^{n^{\varepsilon}}$ running time that fools n^k-size circuits, then $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$.

Setting $S_{\mathsf{ckt}}(n) = n^{\varepsilon}$, $S_{\mathsf{sparse}}(n) = 2^{n^{\beta}}$ and $T(n) = 2^{n^{\beta}}$ in Corollary 6.4, there is an i.o.-NPRG with seed length $O(n^{\beta})$, takes $O(n^{\beta})$ bits of advice, and runs in $2^{O(n^{\beta})}$ time that fools circuits of size $n^{\Omega(\varepsilon/\beta)} = n^{\varepsilon'}$ for $\varepsilon' > 0$. By setting $m = n^{\varepsilon'/k}$, we obtain an i.o.-NPRG with seed/advice length $O(m^{\beta \cdot k/\varepsilon'})$ and running time $2^{O(m^{\beta \cdot k/\varepsilon'})}$, which fools circuits of size m^k. Therefore, by [IKW02], it follows that $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$.

☐

7 Open Problems

We conclude with three interesting open questions stemming from this work.

1. *Are fixed-polynomial circuit lower bounds for NP equivalent to a Karp-Lipton collapse of PH to NP?*

   Formally, is $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ for all k equivalent to ($\mathsf{NP} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{PH} \subset$ i.o.-$\mathsf{NP}_{/n}$)? Recall we showed that similar Karp-Lipton-style collapses do occur, assuming NP circuit lower bounds (e.g., ($\mathsf{PSPACE} \subset \mathsf{P}_{/\mathsf{poly}} \implies \mathsf{PSPACE} \subset$ i.o.-$\mathsf{NP}_{/n}$)), and we showed that $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ implies a type of collapse of AM into NP.

2. It is also a prominent open problem to prove that $\mathsf{ZPP}_{tt}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[n^k]$ for some constant k [DPV18] (that is, prove lower bounds for ZPP with nonadaptive queries to an NP oracle). *Is this lower bound equivalent to a Karp-Lipton collapse of PH?*

   The difficulty is that, assuming $\mathsf{ZPP}_{tt}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[n^k]$, it appears that we may obtain a good simulation of $\mathsf{BPP}_{tt}^{\mathsf{NP}}$, but we presently have no Karp-Lipton Theorem collapsing PH to $\mathsf{BPP}_{tt}^{\mathsf{NP}}$ (indeed, lower bounds for this class are also open). Furthermore, [DPV18] observe that $\mathsf{NP} \subset \mathsf{P}/\operatorname{poly}$ does imply the (small) collapse $\mathsf{BPP}_{tt}^{\mathsf{NP}} = \mathsf{ZPP}_{tt}^{\mathsf{NP}}$; it is unclear how a circuit lower bound against $\mathsf{ZPP}_{tt}^{\mathsf{NP}}$ would aid a further collapse.

3. In light of our Theorem 1.7, *is it possible to show interesting hardness magnification results for non-sparse versions of MCSP (say, $\mathsf{MCSP}[2^m/m^2]$)?*

   Currently, we only know hardness magnification results when the circuit size parameter is $2^{o(m)}$ [OS18, OPS19, MMW19].

References

[Aar06] Scott Aaronson. Oracles are subtle but not malicious. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 340–354, 2006.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[AKSS95] Vikraman Arvind, Johannes Köbler, Uwe Schöning, and Rainer Schuler. If NP has polynomial-size circuits, then ma=am. *Theor. Comput. Sci.*, 137(2):279–282, 1995.

[BCG+96] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BH92] Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, pages 116–127, 1992.

[BS06] Joshua Buresh-Oppenheim and Rahul Santhanam. Making hard problems harder. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 73–87, 2006.

[Cai07] Jin-yi Cai. $\mathsf{S}_2^{\mathsf{p}}$ is subset of $\mathsf{ZPP}^{\mathsf{np}}$. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007.

[CCHO05] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved karp-lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.

[Che19] Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:31, 2019.

[CR06] Venkatesan T Chakaravarthy and Sambuddha Roy. Oblivious symmetric alternation. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 230–241. Springer, 2006.

[CR11] Venkatesan T. Chakaravarthy and Sambuddha Roy. Arthur and merlin as oracles. *Computational Complexity*, 20(3):505–558, 2011.

[CT19] Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits "just beyond" known lower bounds. 2019. To appear in STOC.

[CW04] Jin-yi Cai and Osamu Watanabe. On proving circuit lower bounds against the polynomial-time hierarchy. *SIAM J. Comput.*, 33(4):984–1009, 2004.

[DPV18] Peter Dixon, Aduri Pavan, and N. V. Vinodchandran. On pseudodeterministic approximation algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 61:1–61:11, 2018.

[FS17] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Inf. Comput.*, 256:149–159, 2017.

[FSW09] Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed-polynomial size circuit bounds. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 19–26, 2009.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[HIS85] Juris Hartmanis, Neil Immerman, and Vivian Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):158–181, 1985.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 7:1–7:20, 2018.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[Imp18] Russell Impagliazzo. Personal Communication, 2018.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.

[KL82] Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

[KW98] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[MMW19] Dylan McKay, Cody Murray, and Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. 2019. To appear in STOC 2019.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901, 2018.

[OPS19] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. 2019. To appear in CCC 2019.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 65–76, 2018.

[San09] Rahul Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003.

[Vin05] N. V. Vinodchandran. A note on the circuit complexity of PP. *Theor. Comput. Sci.*, 347(1-2):415–418, 2005.

[Yap83] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983.

A Almost Almost-everywhere $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ Circuit Lower Bounds

Here we provide a proof for Lemma 4.1 for completeness. The proof is based on a similar lemma from [Che19].

A.1 Preliminaries

A crucial ingredient of the proof is a PSPACE-complete language [TV07] satisfying strong reducibility properties, which is also used in the fixed-polynomial lower bounds for $\mathsf{MA}_{/1}$ and $\mathsf{promiseMA}$ [San09], and the recent new witness lemmas for NQP and NP [MW18].

We first define these reducibility properties.

Definition A.1. Let $L : \{0,1\}^* \to \{0,1\}$ be a language, we define the following properties:

- L is *downward self-reducible* if there is a constant c such that for all sufficiently large n, there is an n^c size uniform oracle circuit A such that for all $x \in \{0,1\}^n$, $A^{L_{n-1}}(x) = L_n(x)$.
- L is *paddable*, if there is a polynomial time computable projection Pad (that is, each output bit is either a constant or only depends on 1 input bit), such that for all integers $1 \le n < m$ and $x \in \{0,1\}^n$, we have $x \in L$ if and only if $\mathsf{Pad}(x, 1^m) \in L$, where $\mathsf{Pad}(x, 1^m)$ always has length m.
- L is *same-length checkable* if there is a probabilistic polynomial-time oracle Turing machine M with output in $\{0, 1, ?\}$, such that, for any input x,
  - M asks its oracle queries only of length $|x|$.
  - If M is given L as an oracle, then M outputs $L(x)$ with probability 1.
  - M outputs $1 - L(x)$ with probability at most $1/3$ no matter which oracle is given to it.

  We call M an *instance checker* for L.

Remark A.2. *Note that the paddable property implies that* $\mathsf{SIZE}(L_n)$ *is non-decreasing.*

The following PSPACE-complete language is given by [San09] (modifying a construction of Trevisan and Vadhan [TV07]).

Theorem A.3 ([San09, TV07])**.** *There is a* PSPACE*-complete language* L^{PSPACE} *which is paddable, downward self-reducible, and same-length checkable.*

We also need the following folklore theorem which is proved by a direct diagonalization against all small circuits.

Theorem A.4. *Let* $n \le s(n) \le 2^{o(n)}$ *be space-constructible. There is a universal constant* c *and a language* $L \in \mathsf{SPACE}[s(n)^c]$ *that* $\mathsf{SIZE}(L_n) > s(n)$ *for all sufficiently large* n.

A.2 Definitions

We need the following convenient definition of an $\mathsf{MA} \cap \mathsf{coMA}$ algorithm, which simplifies the presentation.

Definition A.5. A language L is in $\mathsf{MA} \cap \mathsf{coMA}$, if there is a deterministic algorithm $A(x, y, z)$ (which is called the predicate) such that:

- A takes three inputs x, y, z such that $|x| = n$, $|y| = |z| = \mathrm{poly}(n)$ (y is the witness while z is the collection of random bits), runs in $O(T(n))$ time, and outputs an element from $\{0, 1, ?\}$.
- (Completeness) There exists a y such that
$$\Pr_z[A(x, y, z) = L(x)] \ge 2/3.$$
- (Soundness) For all y,
$$\Pr_z[A(x, y, z) = 1 - L(x)] \le 1/3.$$

Remark A.6. ($\mathsf{MA} \cap \mathsf{coMA}$) *languages with advice are defined similarly, with* A *being an algorithm with the corresponding advice.*

Note that by above definition, the semantic of $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ is different from $\mathsf{MA}_{/1} \cap \mathsf{coMA}_{/1}$. A language in $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ has both an $\mathsf{MA}_{/1}$ algorithm and a $\mathsf{coMA}_{/1}$ algorithm, and *their advice bits are the same*. While a language in $\mathsf{MA}_{/1} \cap \mathsf{coMA}_{/1}$ can have an $\mathsf{MA}_{/1}$ algorithm and a $\mathsf{coMA}_{/1}$ algorithm with different advice sequences.

A.3 Proof for Lemma 4.1

Now we are ready to prove Lemma 4.1 (restated below).

Reminder of Lemma 4.1. *For all constants k, there is an integer c, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})_{/1}$, such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*

- *$\mathsf{SIZE}(L_n) > n^k$, or*
- *$\mathsf{SIZE}(L_m) > m^k$, for an $m \in (n^c, 2 \cdot n^c) \cap \mathbb{N}$.*

Proof. Let L^{PSPACE} be the language specified by Theorem A.3. By Theorem A.4, there is an integer c_1 and a language L^{diag} in $\mathsf{SPACE}(n^{c_1})$, such that $\mathsf{SIZE}(L^{\mathsf{diag}}_n) \ge n^k$ for all sufficiently large n. By the fact that L^{PSPACE} is PSPACE-complete, there is a constant c_2 such that L^{diag}_n can be reduced to L^{PSPACE} on input length n^{c_2} in n^{c_2} time. We set $c = c_2$.

The Algorithm. Let $\tau \in \mathbb{N}$ be sufficiently large. We also let b to be a constant to be specified later. Given an input x of length $n = 2^\tau$ and let $m = n^c$, we first provide an informal description of the $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm which computes the language L. There are two cases:

1. When $\mathsf{SIZE}(L^{\mathsf{PSPACE}}_m) \le n^b$. That is, when L^{PSPACE}_m is *easy*. In this case, on inputs of length n, we guess-and-verify a circuit for L^{PSPACE}_m of size n^b and use that to compute L^{diag}_n.
2. Otherwise, we know L^{PSPACE}_m is *hard*. Let ℓ be the largest integer such that $\mathsf{SIZE}(L^{\mathsf{PSPACE}}_\ell) \le n^b$. On inputs of length $m_1 = m + \ell$, we guess-and-verify a circuit for L^{PSPACE}_ℓ and compute it (that is, compute L^{PSPACE}_ℓ on the first ℓ input bits while ignoring the rest).

Intuitively, the above algorithm computes a hard function because either it computes the hard language L^{diag}_n on inputs of length n, or it computes the hard language L^{PSPACE}_ℓ on inputs of length m_1. A formal description of the algorithm is given in Algorithm 1, while an algorithm for setting the advice sequence is given in Algorithm 2. It is not hard to see that a y_n can only be set once in Algorithm 2.

The Algorithm Satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ Promise. We first show the algorithm satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise (Definition A.5). The intuition is that it only tries to guess-and-verify a circuit for L^{PSPACE} when it exists, and the properties of the instance checker (Definition A.1) ensure that in this case the algorithm satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise. Let $y = y_n$, there are three cases:

1. $y = 0$. In this case, the algorithm computes the all zero function, and clearly satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise.
2. $y = 1$ and n is a power of 2. In this case, from Algorithm 2, we know that $\mathsf{SIZE}(L^{\mathsf{PSPACE}}_m) \le n^b$ for $m = n^c$. Therefore, at least one guess of the circuit is the correct circuit for L^{PSPACE}_m, and on that guess, the algorithm outputs $L^{\mathsf{diag}}_n(x) = L^{\mathsf{PSPACE}}_m(z)$ with probability at least $2/3$, by the property of the instance checker (Definition A.1).

   Again by the property of the instance checker, on all possible guesses, the algorithm outputs $1 - L^{\mathsf{diag}}_n(x) = 1 - L^{\mathsf{PSPACE}}_m(z)$ with probability at most $1/3$. Hence, the algorithm correctly computes L^{diag}_n on inputs of length n, with respect to Definition A.5.
3. $y = 1$ and n is not a power of 2. In this case, from Algorithm 2, we know that $\mathsf{SIZE}(L^{\mathsf{PSPACE}}_\ell) \le n_0^b$. Therefore, at least one guess of the circuit is the correct circuit for L^{PSPACE}_ℓ, and on that guess, the algorithm outputs $L^{\mathsf{PSPACE}}_\ell(z)$ ($z = z(x)$ is the first ℓ bits of x) with probability at least $2/3$, by the property of the instance checker (Definition A.1).

Algorithm 1: The $\mathsf{MA} \cap \mathsf{coMA}$ algorithm

Given an input x with input length $n = |x|$;
Given an advice bit $y = y_n \in \{0, 1\}$;
Let $m = n^c$;
Let $n_0 = n_0(n)$ be the largest integer such that $n_0^c \leq n$;
Let $m_0 = n_0^c$;
Let $\ell = n - m_0$;
if $y = 0$ **then**
    Output 0 and terminate
if n *is a power of* 2 **then**
    (We are in the case that $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$.);
    Compute z in n^c time such that $L_n^{\mathsf{diag}}(x) = L_m^{\mathsf{PSPACE}}(z)$;
    Guess a circuit C of size at most n^b;
    Let M be the instance checker for L_m^{PSPACE};
    Flip an appropriate number of random coins, let them be r;
    Output $M^C(z, r)$;
else
    (We are in the case that $\mathsf{SIZE}(L_{m_0}^{\mathsf{PSPACE}}) > n_0^b$ and ℓ is the largest integer such that $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n_0^b$.);
    Let z be the first ℓ bits of x;
    Guess a circuit C of size at most n_0^b;
    Let M be the instance checker for L_ℓ^{PSPACE};
    Flip an appropriate number of random coins, let them be r;
    Output $M^C(z, r)$;

Algorithm 2: The algorithm for setting advice bits

All y_n's are set to 0 by default;
for $\tau = 1 \to \infty$ **do**
    Let $n = 2^\tau$;
    Let $m = n^c$;
    if *SIZE*$(L_m^{\mathsf{PSPACE}}) \leq n^b$ **then**
        Set $y_n = 1$;
    else
        Let $\ell = \max\{\ell : \mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b\}$;
        Set $y_{m+\ell} = 1$;

Again by the property of the instance checker, on all possible guesses, the algorithm outputs $1 - L_\ell^{\mathsf{PSPACE}}(z)$ with probability at most $1/3$. Hence, the algorithm correctly computes $L_\ell^{\mathsf{PSPACE}}(z(x))$ on inputs of length n, with respect to Definition A.5.

The Algorithm Computes a Hard Language. Next we show that the algorithm indeed computes a hard language as stated. Let τ be a sufficiently large integer, $n = 2^\tau$, and $m = n^c$. According to Algorithm 2, there are two cases:

- $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \le n^b$. In this case, Algorithm 2 sets $y_n = 1$. And by previous analyses, we know that L_n computes the hard language L_n^{diag}, and therefore $\mathsf{SIZE}(L_n) > n^k$.
- $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) > n^b$. Let ℓ be the largest integer such that $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \le n^b$. By Remark A.2, we have $0 < \ell < m$.

  Note that $\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}}) \le (\ell+1)^d \cdot \mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}})$ for a universal constant d, because L^{PSPACE} is downward self-reducible. Therefore,

$$\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \ge \mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}})/(\ell+1)^d \ge n^b/m^d \ge n^{b-c\cdot d}.$$

  Now, on inputs of length $m_1 = m + \ell$, we have $y_{m_1} = 1$ by Algorithm 2 (note that $m_1 \in (m, 2m)$ as $\ell \in (0, m)$). Therefore, L_{m_1} computes L_ℓ^{PSPACE}, and

$$\mathsf{SIZE}(L_{m_1}) = \mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \ge n^{b-c\cdot d}.$$

  We set b such that $n^{b-c\dot{d}} \ge (2m)^k \ge m_1^k$ (we can set $b = cd + 3 \cdot ck$), which completes the proof.

$\square$

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il