We will now prove a stronger result, which follows a similar blueprint but whose assumption is considerably weaker. This is the basic form of the original "algorithmic method" of Williams.

Theorem 1 ([Wil10]). *Assume that* $\mathsf{CAPP}_{1,1/2}$ *on n-bit circuits of size* $\mathrm{poly}(n)$ *can be solved in non-deterministic time* $2^n/n^{\omega(1)}$. *Then* $\mathcal{NEXP} \not\subset \mathcal{P}/\mathrm{poly}$.

To parse the assumption, note that the brute-force algorithm takes time $2^n \cdot \mathrm{poly}(n)$. Thus, we only need to shave a large enough polylogarithmic factor from the runtime in order to deduce the lower bound! Recall that if $prBPP = prP$ then $\mathsf{CAPP}_{1,1/2}$ for such circuits can be solved in time $\mathrm{poly}(n)$, let alone in time $2^n/n^{\omega(1)}$.

Going further, a polynomial improvement in the runtime (rather than only polylogarithmic) would lead to stronger conclusions, for example:

Theorem 2 ([MW20]). *Assume that for some* $\epsilon > 0$, $\mathsf{CAPP}_{1,1/2}$ *on n-bit circuits of size* $2^{\epsilon \cdot n}$ *can be solved in time non-deterministic* $2^{(1-\epsilon)\cdot n}$. *Then for every k,* $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.

Here the problem being solved is actually in $\mathcal{P}$, because the given circuit is of size $2^{\epsilon \cdot n}$ and the brute-force algorithm runs in time $2^{(1+O(\epsilon))\cdot n}$. Thus, saving a polynomial runtime factor (i.e., of $2^{O(\epsilon \cdot n)}$) implies $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.

The approach, in general, scales down to weak circuit classes: CAPP algorithms for $\mathcal{C}$ (or for very mild extensions of $\mathcal{C}$) imply lower bounds for $\mathcal{C}$. This was not trivial to prove, and was established in a sequence of follow-up works (see e.g. [BV14; CW19]).

The hallmark success of this approach was proving [Wil11] a lower bound in $\mathcal{NEXP}$, then later even in $\mathcal{NQP}$, for a weak circuit class denoted by $\mathcal{ACC}$ (which we will study in later classes). Doing so was an open problem for several decades.

1 Witness lower bounds

Recall that previous proofs used a case analysis, depending on whether or not there are small circuits computing the prover strategy in a proof system for a hard problem. We'll use a similar notion now, thinking of witnesses in an $\mathcal{NTIME}$ system as strings/truth-tables, and asking whether or not there are small circuits whose truth-table is the witness. Intuitively, we expect the same phenomenon: if witnesses have small circuits, we can guess a circuit for a witness, instead of guessing the entire witness (and hopefully that can lead to some speed-up).

Definition 3. *Let* V *be a non-deterministic* T*-time machine. We say that* V has witnesses of size s *if for every* $x \in L$ *there is* $w \in \{0,1\}^T$ *such that* w *is the truth table of a function* $w\colon \{0,1\}^{\log(T)} \to \{0,1\}$ *that has a circuit of size* $s(\log(T))$.

Note the order of quantifiers: for every x there is an easy w. The reason for this is our target win-win application:

1. If every x has at least one easy witness, then we can guess this easy witness (and hopefully get some speed-up).

2. If there are infinitely many x such that *every convincing witness has high circuit complexity*, this gives us a way to obtain a hard function! Assuming that we have x, we guess w, check that it is accepted by V, and then we know for certain that w is a truth-table of a hard function.

Probabilistically checkable proofs. Recall that the KL-style approach is to start with a hard problem L, and given input x we guess a small "prover-circuit" and try to speed-up the computation of $L(x)$ using this circuit. We now apply this approach to $L \in \mathcal{NTIME}[T]$ and witnesses for L, i.e. we guess a small circuit whose truth-table is (hopefully) a convincing witness w.

The bottleneck, however, is that the $\mathcal{NTIME}[T]$ verifier still needs to read all of w (i.e., the entire truth-table of the guessed circuit), so we did not save much in total runtime. To overcome the bottleneck we'll use an alternative proof system for $\mathcal{NTIME}[T]$, in which the verifier doesn't need to read the entire witness w, but uses randomness instead to choose a small number of bits of w to read.

Definition 4. *A PCP verifier for L is an algorithm V that, for every input x:*

- *If $x \in L$ there is π such that $\Pr_r[V^\pi(x,r) = 1] = 1$.*
- *If $x \notin L$, for every π we have $\Pr_r[V^\pi(x,r) = 1] \leq 1/2$.*

We care about the running time of V, its randomness complexity, and the number of queries it makes to π.

Theorem 5 ([BGH+05])**.** *For every $L \in \mathcal{NTIME}[T]$ there is a PCP verifier with runtime $\mathrm{poly}(n, \log(T))$ (this is also a bound on the number of queries) that uses at most $\log(T) + O(\log\log T)$ random coins (hence $\tilde{O}(T)$ is a bound on proof length).*[1]

Here is the main lemma on the way to proving Theorems 1 and 2, asserting that non-trivial CAPP algorithms imply *witness* lower bounds.

Theorem 6. *Assume that $\mathsf{CAPP}_{1,1/2}$ on n-bit circuits of size $\mathrm{poly}(n)$ can be solved in non-deterministic time $2^n/n^{\omega(1)}$. Then there is a unary $L \in \mathcal{NTIME}[2^{O(n)}]$ with a verifier that does not have polynomial-sized witnesses.*

Proof. Let $L \in \mathcal{NTIME}[2^n] \setminus \mathcal{NTIME}[o(2^n)]$ be unary [Žák83]. Let V be a PCP for L verifier with runtime $\mathrm{poly}(n)$ and randomness $n' = n + O(\log n)$. We'll prove that PCP witnesses for V can't be of polysize. From this we can obtain a standard $\mathcal{NTIME}[2^{O(n)}]$ verifier V' for L that doesn't have polysize circuits (i.e., V' simply enumerates over the random coins of V, so V' accepts (x,w) iff V accepts (x,w)).

Assume tac that PCP witnesses for V are of size n^k, for some k. We will speed up the computation of L, using the following algorithm:

[1] Given recent results about locally testable codes, there is hope to see an improvement of an $O(T)$ bound on the proof length, resolving a long-standing open problem. However, this is immaterial for the current application.

- Given x, guess a circuit $C_w\colon \{0,1\}^{n'} \to \{0,1\}$ of size n^k.
- Construct a circuit $C_{x,C_w}\colon \{0,1\}^{n'} \to \{0,1\}$ such that $C_{x,C_w}(r) = V^{C_w}(x,r)$.
- Run the CAPP algorithm on C_{x,C_w}.

The algorithm above runs in time

$$2^{n'}/n'^{\omega} + \mathrm{poly}(n) \leq 2^n/n^{\omega(1)} \cdot \mathrm{poly}(n) = o(2^n)\ .$$

If $x \in L$ then there is w with a circuit C_w of size n^k such that $\Pr[V^w(x,r)=1]=1$, and when the algorithm guesses C_w it accepts x. Otherwise, for every w – in particular, for every truth-table of a circuit C_w – we have $\Pr[V^w(x,w)=1] \leq 1/2$, hence the algorithm rejects x. ■

The result scales by a simple change of parameters, for example:

Theorem 7. *Assume that* $\mathsf{CAPP}_{1,1/2}$ *on n-bit circuits of size $2^{\epsilon\cdot n}$ can be solved in non-deterministic time $2^{(1-\epsilon)\cdot n}$, for some $\epsilon > 0$. Then there is a unary $L \in \mathcal{NTIME}[2^{O(n)}]$ with a verifier that does not have witnesses of size $2^{\delta\cdot n}$, for some $\delta = \delta(\epsilon) > 0$.*

2 From witness lower bounds to circuit lower bounds

We show how to get from witness lower bounds (i.e., witnesses have high circuit complexity) to circuit lower bounds (i.e., there is a language with high circuit complexity). Note that this is not trivial: witness lower bounds for a unary language imply that, given 1^n, we can guess-and-verify a hard w; however, different guesses may yield different truth-tables. To see why this is a problem, consider trying to define a single hard truth-table L_n. For any $x \in \{0,1\}^n$ it may be the case that some convincing witness w has $w_x = 1$ and another convincing witness has $w_x = 0$.

Instead of the brute-force approach, we will use witness lower bounds to design a non-deterministic PRG, a notion whose definition is implicit in the proof below. Using such a PRG, we will derandomize $\mathcal{MA}$ (and even $\mathcal{MA}/1$) into $\mathcal{NP}$, in which case the lower bound $\mathcal{MA}/1 \not\subset \mathcal{SIZE}[n^k]$ becomes $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$. Details follow.

Definition 8. *A distribution D on $\{0,1\}^n$ ϵ-fools a circuit C if*

$$\left| \Pr_{r\in\{0,1\}^n}[C(r)=1] - \Pr_{r\sim D}[C(r)=1] \right| \leq \epsilon\ .$$

Theorem 9 ([NW94; IW97; Uma03])**.** *There is an algorithm G satisfying the following. Assume that $f \in \{0,1\}^N$ has circuit complexity more than s^c, where $c > 1$ is a universal constant. Then $G(f,1^s)$ runs in time $\mathrm{poly}(N)$ and outputs a list of s-bit strings such that for every circuit $C\colon \{0,1\}^s \to \{0,1\}$ of linear size, the distribution $G(f,1^s)$ $(1/s)$-fools C.*

Theorem 10. *Assume that there is a verifier for some unary $L \in \mathcal{NTIME}[2^{O(n)}]$ that does not have witnesses of size $2^{\delta \cdot n}$, for some $\delta > 0$. Then $\mathcal{MA} \subseteq \text{i.o.}\mathcal{NP}$, and moreover $\mathcal{MA}/1 \subseteq \text{i.o.}\mathcal{NP}/1$.*

Proof. Given x and a witness w of size n^k, create a circuit $C_{x,w}$ of size at most $n^{2k} = s$. With a verifier V_L for L on input $1^{\ell = O_{c,k}(\log n)}$, guess and verify a truth-table of length $N = 2^{O(\ell)}$ with hardness $s^c = n^{2kc} = 2^{\delta \cdot \ell}$. The PRG runs in time $\text{poly}(N) = \text{poly}(n)$, and it fools $C_{x,w}$ on any n such that V_L on input length ℓ has no witnesses of circuit complexity $2^{\delta \cdot \ell}$. (On other input lengths we obtain a non-deterministic machine that decides some language, but perhaps not L.) The "moreover" part following using the same argument. ■

Proving Theorems 1 and 2. Assuming a $\text{CAPP}_{1,1/2}$ algorithm as in the hypothesis of Theorem 2, by Theorem 7 there is a verifier for a unary $L \in \mathcal{NTIME}[2^{O(n)}]$ that doesn't have witnesses of size $2^{\delta \cdot n}$, and thus by Theorem 10 we have $\mathcal{MA}/1 \subseteq \text{i.o.}\mathcal{NP}/1$. We will apply this to the lower bound from [San09] to deduce that $\mathcal{NP}/1 \not\subset \mathcal{SIZE}[n^k]$ for all $k \in \mathbb{N}$, and finally eliminate the advice bit by an elementary argument, to deduce that $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.[2]

There is, however, a major gap in the argument above, which is that the derandomization only worked *infinitely often*; that is, we deduced $\mathcal{MA}/1 \subseteq \text{i.o.}\mathcal{NP}/1$ rather than $\mathcal{MA}/1 = \mathcal{NP}/1$. The trouble is that the lower bound in [San09] also holds only infinitely often. What happens if the infinite set of input lengths on which $L \in \mathcal{MA}/1$ is hard is disjoint from the infinite set of input lengths on which we can simulate L in $\mathcal{NP}/1$? This could be serious trouble, but fortunately, Chen [Che19] showed how to overcome this. Loosely speaking, he showed that the lower bound holds on a sufficiently dense set $S \subseteq \mathbb{N}$ of input lengths, and observed that a non-deterministic PRG that yields derandomization on input length n also yields efficient derandomization on input lengths close to n. In particular, the densities are such that S intersects the set of inputs on which the derandomization works, infinitely many times. ■

3 Recap and comments

Starting with a $\text{CAPP}_{1,1/2}$ algorithm for n-bit circuits of size $2^{\epsilon \cdot n}$ running in time $2^{(1-\epsilon) \cdot n}$, the first two steps (witness lower bounds + NPRGs) imply that $pr\mathcal{BPP} \subseteq \text{i.o.}pr\mathcal{NP}$. Note a "lifting" phenomenon: we start with a slightly non-trivial CAPP algorithm, but end up with a *polytime* CAPP algorithm.[3] How could this be?

The lifting phenomenon exists, but it's not as extreme as going from a tiny improvement over exponential-time to a fully polytime algorithm. This is because the hypothesis is *already* a polynomial-time CAPP algorithm, i.e. getting a circuit of size

[2] Indeed, if $\mathcal{NP}$ has small circuits then $\mathcal{NP}/1$ has small circuits. Convince yourself of this fact, and also note why your argument breaks when working with $\mathcal{MA}$ instead of $\mathcal{NP}$.

[3] Note that the derandomization at the end is non-deterministic, but the derandomization we start from may also be non-deterministic, so that is not a loss.

$2^{\epsilon \cdot n}$ and running in time $2^{(1-\epsilon)\cdot n}$. The lifting part is that we deduce a polytime CAPP algorithm for all circuits from a polytime CAPP algorithm that only works for circuits with *logarithmically many input bits* (i.e., n-bit circuits of size $2^{\epsilon \cdot n}$); indeed, to do so we need a fine-grained runtime bound on the latter algorithm.

3.1 Historical development

The original proof of the algorithmic method in [Wil10] is structured differently than the one above, originally using a notion of an "easy witness lemma" (introduced in [IKW02] and improved in [MW20]). The "derandomization-centric" perspective presented above is due to Chen [Che19; Che23], following ideas from [Wil16].

Also, scaling down the proof to weak circuit classes is non-trivial. Early versions (including [Wil11; SW13]) used ad-hoc arguments, and a more general argument was given by Ben-Sasson and Viola [BV14], who showed a PCP with a verifier that can be implemented extremely efficiently as a circuit (so the size and depth overhead added in the proof is minimal). An extreme optimization of circuitry overheads, using PCPPs, was shown by Chen and Williams [CW19].

Remark 11. *A direct precursor to the algorithmic method is [IKW02], who showed that if $\mathcal{MA} \neq \mathcal{NEXP}$ (think of this assumption as "weak derandomization") then $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$. In fact, the argument in [IKW02] is similar to the proofs of Theorems 1 and 2, since both of them use essentially the same algorithm (i.e., applying the hypothsized* CAPP *algorithm to an $\mathcal{MA}$ verifier that guesses a small circuit and uses it to compute a PCP witness) to speed-up computation of any $\mathcal{NEXP}$ language, and contradict an $\mathcal{NTIME}$-hierarchy. However, this "sped-up" algorithm and its use are opaque in the text of [IKW02], which organizes the proof differently,[4] and the parameters obtained in the final statement are not nearly as tight.*

3.2 Extensions

The algorithmic method was extended to imply average-case circuit lower bounds. Showing this is highly non-trivial, especially for weak circuit classes (that cannot compute error-correcting codes), and these results were developed in a sequence of works [COS18; Che19; CL21; CLL+21; Che22]. We'll see this in the final presentations.

Another strengthening of the conclusion deduces lower bounds in which every circuit family fails on all input lengths (except, at most, finitely many), rather than only infinitely often. We will see this result in two lectures.

[4] Inspecting the proof of the relevant direction in [IKW02, Theorem 32], it works as follows. They first show that $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$ implies that $\mathcal{NEXP} = \mathcal{EXP}$ (see [IKW02, Theorem 24]). Now, assuming a CAPP algorithm and hence derandomization of $\mathcal{MA}$, they use the assumption $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$ to speed up the computation of any $\mathcal{NEXP}$ language, and obtain a contradiction to an $\mathcal{NTIME}$-hierarchy. Specifically, in [IKW02, Theorems 32] they apply the CAPP algorithm to an $\mathcal{MA}$ verifier that is essentially identical to the one in the proofs of Theorems 1 and 2, except that it uses the PCP of [BFL91] (this $\mathcal{MA}$ verifier is the one used to deduce that $\mathcal{NEXP} = \mathcal{MA}$ in [IKW02, Theorem 23], which is indeed just the $\mathcal{MA}$ verifier from [BFL91, Corollaries 6.9 & 6.10]), and deduce a contradiction to a specific $\mathcal{NTIME}$-hierarchy that holds under the assumption (in [IKW02, Corollary 9]). This organization of their proof is not transparent in the original text, but rather a retrospective view in light of [Wil10].

A strengthening of the conclusion that we *don't* know refers to circuits of larger size. In Theorem 2 we can deduce that $\mathcal{NTIME}[t] \not\subset \mathcal{SIZE}[s]$ for $t = \text{poly}(s(\text{poly}(s))) \approx s \circ s$, because the lower bound of [San09] is of this form; this gives non-trivial lower bounds for circuits of "half-exponential" size (see, e.g., [Tel19; Che23]). An outstanding open question is whether we can deduce non-trivial lower bounds for circuits of exponential size $2^{\epsilon \cdot n}$, which would bring us very close to an equivalence between the algorithmic assumption and the lower bound conclusion.

A different direction is relaxing the hypothesis. We know that for any $B(n) \leq 2^n/3$, a CAPP algorithm distinguishing between n-bit circuits that accept all their inputs and circuits that reject all but at most B of their inputs in time $B \cdot (\log B)^{\omega(1)}$ implies circuit lower bounds (this setting is called "quantified derandomization", see [GW13; Tel22]).

References

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. "Non-Deterministic Exponential Time has Two-Prover Interactive Protocols". In: *Comput. Complex.* 1 (1991), pp. 3–40.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. "Short PCPs Verifiable in Polylogarithmic Time". In: *Proc. 20th Annual IEEE Conference on Computational Complexity (CCC)*. 2005, pp. 120–134.

[BV14] Eli Ben-Sasson and Emanuele Viola. "Short PCPs with projection queries". In: *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*. 2014, pp. 163–173.

[Che19] Lijie Chen. "Non-deterministic Quasi-Polynomial Time is Average-case Hard for ACC Circuits". In: *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2019, pp. 1281–1304.

[Che22] Lijie Chen. "Better Hardness via Algorithms, and New Forms of Hardness versus Randomness". PhD thesis. Massachusetts Institute of Technology, 2022.

[Che23] Lijie Chen. "New lower bounds and derandomization for *ACC*, and a derandomization-centric view on the algorithmic method". In: *Proc. 14th Conference on Innovations in Theoretical Computer Science (ITCS)*. Vol. 251. LIPIcs. Leibniz Int. Proc. Inform. 2023, Art. No. 34, 15.

[CL21] Lijie Chen and Xin Lyu. "Inverse-exponential correlation bounds and extremely rigid matrices from a new derandomized XOR lemma". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 761–771.

[CLL+21] Lijie Chen, Zhenjian Lu, Xin Lyu, and Igor C. Oliveira. "Majority vs. Approximate Linear Sum and Average-Case Complexity Below NC^1". In: *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*. 2021, 51:1–51:20.

[COS18] Ruiwen Chen, Igor Carboni Oliveira, and Rahul Santhanam. "An Average-Case Lower Bound Against ACC^0". In: *Proc. 13th Theoretical Informatics - Latin American Symposium (LATIN)*. 2018, pp. 317–330.

[CW19] Lijie Chen and R. Ryan Williams. "Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity". In: *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 2019, 19:1–19:43.

[GW13] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), p. 152.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In search of an easy witness: exponential time vs. probabilistic polynomial time". In: *Journal of Computer and System Sciences* 65.4 (2002), pp. 672–694.

[IW97] Russell Impagliazzo and Avi Wigderson. "$\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: derandomizing the XOR lemma". In: *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 1997, pp. 220–229.

[MW20] Cody D. Murray and R. Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-polytime from a New Easy Witness Lemma". In: *SIAM Journal on Computing* 49.5 (2020).

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[San09] Rahul Santhanam. "Circuit lower bounds for Merlin-Arthur classes". In: *SIAM Journal on Computing* 39.3 (2009), pp. 1038–1061.

[SW13] Rahul Santhanam and R. Ryan Williams. "On medium-uniformity and circuit lower bounds". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. IEEE, 2013, pp. 15–23.

[Tel19] Roei Tell. "Proving that $\mathsf{prBPP} = \mathsf{prP}$ is as hard as proving that "almost NP" is not contained in $\mathsf{P/poly}$". In: *Information Processing Letters* 152 (2019), p. 105841.

[Tel22] Roei Tell. "Quantified derandomization: how to find water in the ocean". In: *Foundations and Trends® in Theoretical Computer Science* 15.1 (2022), Paper No 1, 125.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *Journal of Computer and System Sciences* 67.2 (2003), pp. 419–440.

[Wil10] Ryan Williams. "Improving exhaustive search implies superpolynomial lower bounds". In: *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*. 2010, pp. 231–240.

[Wil11] Ryan Williams. "Non-uniform ACC circuit lower bounds". In: *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*. 2011, pp. 115–125.

[Wil16] R. Ryan Williams. "Natural Proofs versus Derandomization". In: *SIAM Journal on Computing* 45.2 (2016), pp. 497–529.

[Žák83] Stanislav Žák. "A Turing machine time hierarchy". In: *Theoretical Computer Science* 26.3 (1983), pp. 327–333.