Our goal today and in some of next week will be to prove the following result, which is the starting point of our journey for "algorithms ⇒ lower bounds":

Theorem 1. *Assume that there is a deterministic polytime algorithm that gets as input a circuit $C\colon \{0,1\}^n \to \{0,1\}$ and outputs "yes" if $\Pr_r[C(r)=1]=1$, and outputs "no" if $\Pr_r[C(r)=1] \leq 1/2$. Then, for every $k \in \mathbb{N}$ we have $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.*

The problem in the assumption is usually called Circuit Acceptance Probability Problem (CAPP), which can be defined with any two thresholds; the version above is a "one-sided error" version with thresholds 1 and 1/2, denoted $\mathsf{CAPP}_{1,1/2}$. This problem can be trivially solved using randomness. Deterministically, it's complete for $prBPP$ under polytime reductions, and hence the assumption above is equivalent to $prBPP = prP$. We have reason to suspect that it's true, since it follows from strong enough circuit lower bounds (e.g., from $\mathcal{E} \not\subset \mathtt{i.o.}\mathcal{SIZE}[2^{.01\cdot n}]$).

Turning to the conclusion, note that it is different from $\mathcal{NP} \not\subset \mathcal{P}/\mathrm{poly}$, because the upper-bound uses more time than n^k. Compare this with hierarchy theorems, and note that in our case the upper-bound is even allowed non-determinism. Ideally, with such runtime advantage, we could even hope for better, e.g. that $\mathcal{P} \not\subset \mathcal{SIZE}[n^k]$ for all $k \in \mathbb{N}$ (for example, think of k-clique for a large k).

1 A lower bound in Σ_2

Theorem 2 ([Kan84])**.** *For every $k \in \mathbb{N}$, there is $L \in \Sigma_2$ that is hard for $\mathcal{SIZE}[n^k]$.*

We first begin with a weaker and easier lower bound:

Theorem 3. *For every $k \in \mathbb{N}$, there is $L \in \Sigma_3$ that is hard for $\mathcal{SIZE}[n^k]$.*

Proof. Recall that there is a function on $\ell = \log(a \cdot n^k \cdot \log(n))$ variables that is hard for circuits of size n^k, for some $a > 1$. (This is because the circuit complexity of a random ℓ-bit function is $\Theta(2^\ell/\ell) = \Theta(n^k)$.) Let $f\colon \{0,1\}^\ell \to \{0,1\}$ be the lex-first such function. Then L is defined as

$$x \in L \iff x_{\ell+1} = \ldots = x_n = 0 \wedge f(x_1,\ldots,x_\ell) = 1\ .$$

To decide L, given x, we use quantifiers to check if "the lex-first function on $x_1,\ldots,x_\ell$ with circuit complexity more than n^k accepts $x_1,\ldots,x_\ell$"; that is, we check if

$$\begin{array}{ll}
\exists f \in \{0,1\}^{2^\ell} & \\
\forall C \text{ of size } \leq n^k & \forall f' \in \{0,1\}^{2^\ell} \text{ st } f' < f \\
\exists C' \text{ of size } \leq n^k & \\
\quad \text{such that} & \\
f(x)=1, \quad x_{\ell+1} = \ldots = x_n = 0 & \\
\exists y \in \{0,1\}^\ell \text{ st } C(y) \neq f(y) & \forall y \in \{0,1\}^\ell,\ C'(y) = f'(y)
\end{array}$$

The quantifier over C ensures that f has circuit complexity more than n^k, and the quantifiers over f' and C' ensure that any truth-table lex-earlier than f has circuit complexity at most n^k. The quantifiers over $y \in \{0,1\}^\ell$ can be checked by enumeration in time $2^\ell = \text{poly}(n^k)$ (rather than using another alterating quantifier). ■

We will improve the upper-bound to Σ_2 using proof systems and a case analysis. We will need the classical result of Karp and Lipton, which we've mentioned before.

Theorem 4 ([KL82]). $\mathcal{NP} \subset \mathcal{P}/\text{poly} \Rightarrow \Sigma_2 = \Pi_2 \Rightarrow \forall i \geq 2,\ \Sigma_2 = \Sigma_i = \Pi_i$.

Proof. Under the assumption, we decide Π_2 in Σ_2. Consider any $L \in \Pi_2$,

$$x \in L \iff \forall y \exists z : V(x,y,z) = 1 \,.$$

Think of this as a two-player game: The y-player moves first, and the z-player needs to answer the challenge. Even better, think of this as a weird proof system: The verifier sends all challenges y simultaneously, and it is satisfied if and only if for every challenge y the prover can answer by a convincing z.

The crucial observation is that the prover strategy is computable in $\mathcal{P}^{\mathcal{NP}}$. More accurately, if $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, then there is a polysized circuit that, given (x,y), produces a convincing z if such z exists.[1]

To decide L we will *guess* such a circuit, and then instead of interacting with an actual prover, we (sit alone in our room with the circuit we guessed and) give all challenges to the circuit that we have. That is, if $\mathcal{NP} \subset \mathcal{P}/\text{poly}$ then the prover strategy is computable by a circuit of size n^c, for some c; then,

$$x \in L \iff \exists C \text{ of size } n^c : \forall y, V(x,y,C(x,y)) = 1 \,.$$

We will never accept $x \notin L$ (because V is a sound verifier), and by the assumption, for every $x \in L$, there is some guess of C such that for all y the circuit $C(x,y)$ will produce a convincing z. ■

Proof of Theorem 2. We now prove Theorem 2 by a win-win argument. If $\mathcal{NP} \not\subset \mathcal{P}/\text{poly}$, we are done. Otherwise, $\mathcal{NP} \subset \mathcal{P}/\text{poly} \Rightarrow \Sigma_2 = \Pi_2$, hence $\Sigma_2 = \Sigma_3$, so the hard function in Σ_3 is also computable in Σ_2. ■

What I find amazing in this proof is how complexity-theoretic it is. We are dealing with graphs, but we went through proof systems and win-win arguments.

2 A lower bound in $\mathcal{MA}/1$

Can we get a lower bound with less than two alterations? The answer is roughly yes, with some caveats. Recall that $\mathcal{MA}$ is the randomized version of $\mathcal{NP}$.

[1] This uses the standard search-to-decision reduction for $\mathcal{NP}$-complete problems.

Definition 5. *$L \in \mathcal{MA}$ if there is a ppt verifier V such that*

- *(Completeness.)* $x \in L \Rightarrow \exists w, \Pr_r[V(x,w,r) = 1] = 1$.
- *(Soundness.)* $x \notin L \Rightarrow \forall w, \Pr_r[V(x,w,r) = 1] \leq 1/2$.

Extending Kannan's result, Santhanam showed a lower bound in $\mathcal{MA}/1$.

Theorem 6 ([San09])**.** *For every $k \in \mathbb{N}$, there is $L \in \mathcal{MA}/1$ that is hard for $\mathcal{SIZE}[n^k]$.*

Without the one bit of advice, this would be a strict improvement over Theorem 2 for size-n^k circuits (since $\mathcal{MA} \subset \Sigma_2$, see e.g. [GZ11]). If we allow the hard function to be a promise-problem rather than a language, then we do not need the advice.

To prove this theorem we will imitate the same proof strategy in Kannan's result, with different technical tools. Specifically, we will:

- Show a suitable proof system
- ... use it for a KL-type result
- ... then use the KL-type result in a win-win argument.

2.1 $\mathcal{IP} = \mathcal{PSPACE}$

We will need heavier complexity-theoretic tools. We define interactive proof systems. Intuitively, we model a verifier and a prover that get a shared input x, and interact – the verifier sends a challenge, the prover sends a reponse, the verifier sends another challenge, and so on. Formally, the prover and the verifier are functions, mapping partial transcripts to the next message, and we can ask about the functions' computational complexity. We denote by $\langle V, P \rangle (x)$ the output of V after interacting with P on common input x. Various trivialities can be handled wlog (e.g., what happens if P sends something too long, or goes silent), I won't spell them out.

Definition 7. *$L \in \mathcal{IP}$ if there is a ppt verifier V and a (possibly inefficient) honest prover function P such that*

- *(Completeness.)* $x \in L \Rightarrow \Pr_r[\langle V, P \rangle (x) = 1] = 1$.
- *(Soundness.)* $x \notin L \Rightarrow \forall P^*, \Pr[\langle V, P^* \rangle (x) = 1] \leq 1/2$.

In $\mathcal{MA}$ we had one message, now we have an unlimited number of messages. Does this help?

Theorem 8. *$\mathcal{IP} = \mathcal{PSPACE}$. Moreover, there is a proof system for $\mathcal{PSPACE}$ in which the honest prover's strategy is computable in $\mathcal{PSPACE}$.*

Proof Idea. I'll present one proof idea, for a simpler result $\#\mathcal{P} \subseteq \mathcal{IP}$. (See the exposition on my webpage.) ■

A corollary: Instance checkers. An instance checker is a proof system wherein the prover isn't adaptive (so dishonest provers' hands are tied) but the honest prover only answers queries to the same problem itself we are trying to solve.

Definition 9. *An instance checker for L is a ppt oracle machine M satisfying, for all x:*

- *(Completeness.)* $\Pr[M^L(x) = L(x)] = 1$.
- *(Soundness.)* $\forall L'$, $\Pr[M^{L'}(x) \notin \{L(x), \perp\}] \leq 1/2$.

Intuitively, when there is an $\mathcal{IP}$ for L, we can ask what is the complexity of the (honest) prover's strategy function. In the proof above, it is in $\mathcal{PSPACE}$. In particular, when L is a $\mathcal{PSPACE}$-complete language, we can compute the honest prover's strategy function with queries to L itself (since L is complete).

Lemma 10. *There is a $\mathcal{PSPACE}$-complete L that has an instance checker. Moreover, on inputs x the instance checker only issues oracle queries of length $|x|$.*

The only additional work required to prove the lemma is to ensure same-length queries, and this is done by appropriate padding.

2.2 Proof

For simplicity, we prove a version in which there are $\log(n)$ bits of advice; this presents the main idea. The proof is by a case analysis.

Case 1: $\mathcal{PSPACE} \subset \mathcal{SIZE}[n^k]$. Then $\mathcal{PSPACE} = \mathcal{MA}$; I'll leave this proof as exercise (hint: as in the Karp-Lipton theorem, guess a circuit implementing the strategy of the honest prover in $\mathcal{IP} = \mathcal{PSPACE}$). Since $\Sigma_2 \subseteq \mathcal{PSPACE} = \mathcal{MA}$, by Kannan's theorem we are done.

Case 2: $\mathcal{PSPACE} \not\subset \mathcal{SIZE}[n^k]$. Then there is an infinite set $S \subseteq \mathbb{N}$ of input lengths such that $size_L(m) > m^k$ for every $m \in S$. We define a padded version of L, whose circuit complexity is "just above" n^k; specifically, for every $m \in S$, we define

$$n = m + p \text{ where } p \text{ is such that } size_L(m) \in [n^k + 1, n^{k^2}]$$
$$L'_n = \{x0^p : x \in L_m\}$$

where on other input lengths L'_n is defined trivially.[2]

Clearly $L' \notin \mathcal{SIZE}[n^k]$, because for any n as above deciding L'_n takes size more than n^k (i.e., a circuit for L'_n of size less than $n^k + 1$ yields a circuit for L_m of size less than $size_L(m) \geq n^k + 1$).

[2]Low-level technical issues: There could be many such n's for each m; we choose one arbitrarily. And two m's might collide into the same n, in which case we just discard one of the m's.

For the upper bound, intuitively, we can decide L' in $\mathcal{MA}$, by guessing a circuit of size n^{k^2} and running the instance checker (again, as in the Karp-Lipton theorem). But now I won't leave it as an exercise, because there is one non-trivial issue. Given $x \in \{0,1\}^n$, we need to know if L'_n is trivial or or not, and if it is not, then what is $m = m(n)$. This is done with $\log(n)$ bits of advice (to specify m). Now that we can parse the input as $x0^p$ where $x \in \{0,1\}^m$, we can guess a circuit for L_m of size n^{k^2} and run the instance checker on x (analysis left as exercise).

By being a bit more clever, we can get this down to one bit of advice. (Essentially, instead of specifying $m = m(n)$ exactly, we always set the padding to be a power of two larger than m; so after modding out the largest power of two, we know what m is. The one bit only specifies if L'_n is defined trivially or not.)

Remark 11. *What happens if we don't give advice at all? Note that the $\mathcal{MA}$ algorithm guesses a circuit of size n^{k^2} and uses it as a prover in an instance checker. For some input lengths, there may not be a circuit of size n^{k^2} deciding L (i.e., simulating the prover perfectly), in which case for some $x \in L$ it might be that the maximal acceptance probability of the verifier, over all guessed circuits, is above $1/2$ but less than 1 (say, .51). In this case x is neither accepted nor rejected in terms of the definition of $\mathcal{MA}$.*

Remark 12. *What is the actual hard problem? In one case, the hard problem comes from diagonalization, and in another case the hard problem is unspecified and comes from the assumption.*

3 Proof of the main theorem

Recall the theorem:

Theorem 13. *If $\mathsf{CAPP}_{1,1/2} \in pr\mathcal{P}$, then for every $k \in \mathbb{N}$, $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.*

Proof. Fix k, and let $L = L_k \in \mathcal{MA}/1 \setminus \mathcal{SIZE}[n^{2k}]$. Let V be an $\mathcal{MA}/1$ verifier for L. For every (x,w), given the correct bit of advice b, either $\Pr_r[V(x,w,r)/b = 1] = 1$ or $\Pr_r[V(x,w,r)/b = 1] \leq 1/2$.

Let V' be a non-deterministic machine not using random coins (i.e., $\mathcal{NP}$-type) that gets as input (x,b), guesses w of a suitable length for V, creates a circuit $C_{x,w,b}(r) = V(x,w,r)/b$, and runs the CAPP algorithm on $C_{x,w,b}$. Note that:

- When b is the correct bit of advice, if $x \in L$ then there is w such that $V'(x,w,b) = 1$; and if $x \notin L$ then for all w we have $V'(x,w,b) = 0$.
- On every input (x,b) the verifier V' is a non-deterministic machine that either accepts (for some w) or rejects (for all w).

Define $L' = \{(x,b) : V'(x,b) = 1\}$, and note that $L' \in \mathcal{NP}$. Assume towards a contradiction that $L' \in \mathcal{SIZE}[n^k]$, and let $C = \{C_n\}$ be a circuit family deciding L'. By hard-wiring b we obtain circuits of size at most n^{2k} (in terms of the new input length, which is smaller by 1 after hard-wiring b) deciding L. ■

References

[GZ11] Oded Goldreich and David Zuckerman. "Another Proof That BPP subset PH (and More)". In: ed. by Oded Goldreich. Vol. 6650. Lecture Notes in Computer Science. 2011, pp. 40–53.

[Kan84] Ravindran Kannan. "Towards separating nondeterminism from determinism". In: *Mathematical Systems Theory. An International Journal on Mathematical Computing Theory* 17.1 (1984), pp. 29–45.

[KL82] Richard M. Karp and Richard J. Lipton. "Turing machines that take advice". In: *L'Enseignement Mathématique. Revue Internationale. 2e Série* 28.3-4 (1982), pp. 191–209.

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