Oded (January 27, 2023): The algorithmic method for circuit lower bounds[1]

Preliminaries. The class $\mathcal{ACC}$ consists of Boolean functions computable by families of circuits of polynomial size and constant depth having unbounded fan-in `AND`, `OR`, `NOT`, and `MOD` m gates, where $m > 1$ is an arbitrary constant. In other words, $\mathcal{ACC} = \bigcup_{m \in \mathbb{N}} \mathcal{AC}_0[m]$, where $\mathcal{AC}_0[m]$ refers to (polynomial-size, constant-depth) circuits having unbounded fan-in `AND`, `OR`, `NOT`, and `MOD` m gates. (Indeed, $\mathcal{AC}_0 = \mathcal{AC}_0[1]$, which effectively means that no modular gates are used, whereas $\mathcal{AC}_0[2]$ allows `XOR` gates.)

General background. When discussing circuit (size and/or depth) lower bounds, the point is obtaining them for *explicit* functions; in contrast, it is trivial to get such lower bounds for non-explicit functions or even for functions of high (uniform) time complexity (i.e., just let the algorithm try all functions and all circuits). The question is what is "explicit" and the answer is undetermined; actually, one often wants things to be *as explicit as possible*. Still, the first choice would be that *explicit* means computable in polynomial-time. Often (and also in some places in my *introduction to complexity* book), one may require even more; e.g., computability in log-space. In the current context (i.e., aiming at lower bounds for $\mathcal{ACC}$), we shall settle for less (e.g., *explicit* means computable in exponential time with oracle to $\mathcal{NP}$); needless to say, this timid goal is only due to our limitations (i.e., what we know).[2]

Likewise, almost no "derandomization" (of classes of circuits) is known to hold *unconditionally* (i.e., without assuming lower bounds), where here *derandomization* means deterministic procedures that approximate the fraction of inputs that cause a given circuit to evaluate to 1.[3] Essentially, the only exceptions are:

1. Derandomizations that follow from known circuit lower bounds (e.g., for $\mathcal{AC}_0$).
2. Trivial derandomization (e.g., $\mathcal{BPP}$ is in $\mathcal{EXP}$).

In continuation to (1), we mention that various *conjectured* circuit lower bound imply non-trivial derandomization, and that partial converses of these are known; specifically, some *conjectured* derandomizations imply circuit lower bounds that are not known unconditionally. The result we shall see goes in this converse direction, except that it presents a new (alas very weak) derandomization and infers from it new circuit lower bounds.

Overview. We shall prove that circuits of a certain class, denoted $\mathcal{C}$, cannot compute some explicit functions (i.e., functions in a uniform complexity class $\mathcal{X}$) by using an algorithm for the "analysis" of circuits in a related class, denoted $\widehat{\mathcal{C}}$. By *analysis of circuits* we mean either deciding whether a given circuit computes a constant function (see Take 1) or approximating the probability that the circuit evaluates to 1 (on a random input to the circuit, see Take 2). While the framework

[1] Notes for a presentation to a group of graduate students at Weizmann Institute of Science.

[2] Subsequent works obtained stronger notions of explicitness (e.g., being computable in "quasi-NP"). Here we use a weaker notion of explicitness, since it allowed us to present a simpler proof while still showing many of the interesting/fundamental ideas.

[3] In the case of complexity classes such as $\mathcal{BPP}$, this refers to the residual circuits that describe computation of a randomized algorithm on a fixed input. We mention that non-trivial derandomizations of *specific* randomized algorithms are known. The same hold also for derandomization of space complexity classes such as $\mathcal{BPL}$, but one may argue that these use implicit lower bounds on the power of read-once bounded-space machines.

is generic (i.e., it is independent of $\mathcal{C}$), we can benefit from it only if we have a suitable algorithm for the analysis of circuits in $\widehat{\mathcal{C}}$. Hence, at the very end, one uses $\mathcal{C} = \mathcal{ACC}$, and an adequate algorithm for $\widehat{\mathcal{C}} = \mathcal{ACC}$. (Likewise, $\mathcal{X}$ is determined by the proof of fact (3) below.)

Specifically, assuming towards the contradiction that $X \subseteq \mathcal{C}$, we show that witnesses for membership in any NTime(2^n)-sets have concise representations as circuits in $\mathcal{C}$, which (combined with the hypothesized circuit-analysis algorithm) allows us to speed-up the corresponding NTIME computation. Since the latter task is impossible (by the NTIME hierarchy theorem), we derive a contradiction to $X \subseteq \mathcal{C}$.

Take 1. Recall that proving a circuit lower bound for a Boolean function f means proving that a certain class of circuits cannot compute f. Hence, for a class of circuits, denoted $\mathcal{C}$, our aim is proving that there is an explicit function f that is not in $\mathcal{C}$, where being explicit means that f is in a uniform complexity class $\mathcal{X}$ (e.g., $\mathcal{E}^{\mathcal{NP}}$). The proof will rely on a design of a modestly efficient algorithm for the analysis of circuits in a related class, denoted $\widehat{\mathcal{C}}$. Specifically, we shall use an algorithm that, on input a circuit in the class $\widehat{\mathcal{C}}$, determines whether or not the circuit computes the identity 1 function. Assuming that (n-bit input) circuits in $\mathcal{C}$ can be described by strings of length at most $2^{n/4}$, the (lower bound) proof uses the following facts:

1. A hierarchy theorem for non-deterministic time; specifically, the fact that NTime($2^n/n^{\omega(1)}$) is strictly contained in NTime(2^n).

   Assuming towards the contradiction that $\mathcal{X} \subseteq \mathcal{C}$, we shall derive a contradiction by starting with any $S \in \text{NTime}(2^n)$, and showing (at the end) that $S \in \text{NTime}(2^n/n^{\omega(1)})$.

2. The fact that the non-deterministic computation for deciding membership of x in S can be represented by a 3CNF formula V_x of size $N' \stackrel{\text{def}}{=} \widetilde{O}(2^{|x|})$ that takes N' inputs (representing the non-deterministic choices). Furthermore, on input x and j, the j^{th} clause of V_x (i.e., the literals fed to it) can be determined in polynomial-time.[4]

   (Indeed, V_x represent the verification that the N'-bit input constitutes a witness for $x \in S$; that is, $x \in S$ if and only if there exists $w \in \{0,1\}^{N'}$ such that $V_x(w)=1$.)

3. The fact (proved below for $\mathcal{X} = \mathcal{E}^{\mathcal{NP}}$) that $\mathcal{X}$ contains a function $A : \{0,1\}^* \times \mathbb{N}' \to \{0,1\}$ such that for every $x \in S$ the function $A_x : [N'] \to \{0,1\}$ defined as $A_x(i) \stackrel{\text{def}}{=} A(x,i)$ (for every $i \in [N']$) represents an assignment that satisfies V_x (i.e., $V_x(A_x(1), ..., A_x(N')) = 1$).

   For $\mathcal{X} = \mathcal{E}^{\mathcal{NP}}$ an adequate function A_x can be computed by determining its values sequentially. Specifically, $A_x(i)$ is set to 0 if and only if the i-bit long string $w' \stackrel{\text{def}}{=} A_x(1) \cdots A_x(i-1) \cdot 0$ is a prefix of an N'-bit long string that satisfies V_x (i.e., $\exists w''$ such that $V(x, w'w'') = 1$). Hence, the relevant NP-query is $(x, w'10^{N'-i})$.

4. Next, assuming towards the contradiction that $\mathcal{X} \subseteq \mathcal{C}$, it follows that the function A can be computed by a family of circuits, denoted $\{C_m\}_{m \in \mathbb{N}}$, that is in $\mathcal{C}$. For every $x \in \{0,1\}^n$, we let $C'_x(i) = C_{n+n'}(x,i)$, where $n' = \log_2 N' = \log_2 \widetilde{O}(2^n) = n' + O(\log n)$, and $i \in [N']$.

[4] Recall that it is easy to emulate the computation of a (non-deterministic) machine that runs in $t(n)$-time by a (highly-uniform) circuit of size $O(t(n)^2)$. Here, we use a stronger result asserting that such emulation is possible by (highly-uniform) circuits of size $\widetilde{O}(t(n))$. Furthermore, these circuits 3CNF formulas and they can be constructed by uniform $\mathcal{AC}_0$-circuits.

Recall that V_x is a 3CNF formula, and so $V_x(y)$ can be written as

$$\bigwedge_{j\in[N']} v_{x,j}(y_{i^1_{x,j}}, y_{i^2_{x,j}}, y_{i^3_{x,j}}),$$

where $v_{x,j}$ and the $i^k_{x,j}$'s can be computed in polynomial time when given x and j. Hence, the question of whether C'_x describes an assignment that satisfies V_x translates to the question of whether the circuit $\widehat{c}_x : [N'] \to \{0,1\}$ defined by

$$\widehat{c}_x(j) \stackrel{\text{def}}{=} v_{x,j}\left(C'_x\left(i^1_{x,j}\right), C'_x\left(i^2_{x,j}\right), C'_x\left(i^3_{x,j}\right)\right)$$

is identically 1.

5. Lastly, assuming that we can decide in time $2^{|x|}/|x|^{\omega(1)}$ whether the circuit $\widehat{c}_x : [N'] \to \{0,1\}$ is identically 1, we conclude that S is in $\mathrm{NTime}(2^n/n^{\omega(1)})$, by presenting a non-deterministic machine that, on input $x \in \{0,1\}^n$, first guesses $C_{n+n'}$, by making $2^{(n+n')/4} < 2^n/n^{\omega(1)}$ non-deterministic steps[5], then constructs $\widehat{c}_x$, and finally invokes the foregoing decision procedure on $\widehat{c}_x$. Note that the circuit $\widehat{c}_x$ is obtained by composing three copies of $C'_x = C_{n+n'}(x,\cdot)$ with circuits that compute the mapping $j \mapsto v_{x,j}$ and $j \mapsto i^k_{x,j}$ (for $k = 1,2,3$). Furthermore, given x, the circuits that compute these mappings can be constructed in polynomial-time. Actually, these circuits are of the $\mathcal{AC}_0$-type (see Footnote 4).

Indeed, for the foregoing to yield $S \in \mathrm{NTime}(2^n/n^{\omega(1)})$, we need an algorithm as postulated in the last item. Specifically, defining $\widehat{\mathcal{C}}$ as the class of circuits obtained by a constant number of composition operators applied to circuits in $\mathcal{C}$ and $\mathcal{AC}_0$, and recalling that $n' = n + O(\log n)$, we need an $2^n/n^{\omega(1)}$-time algorithm that given an n'-bit circuit in $\widehat{\mathcal{C}}$ determines whether the circuit computes the identity 1 function.

Note that the algorithm we seek is faced with an $\mathcal{NP}$-type problem (i.e., determining whether a given circuit evaluates to 1 on all inputs), but this task refers to a restricted class of circuits (i.e., $\widehat{\mathcal{C}}$) and the algorithm is merely required to be better than exhaustive search in an extremely modest manner (when the circuit is of polynomial size). An even more modest task is to distinguish between the case that circuit evaluates to 1 on all inputs and the case that circuit evaluates to 1 on at most half of the inputs. The latter task is of a derandomization flavor (since it can be easily performed by a randomized algorithm). This leads to –

Take 2. Again, our aim is proving that some function $f \in \mathcal{X}$ is not in $\mathcal{C}$, and the proof will rely on a design of a modestly efficient algorithm for the analysis of circuits in a related class, denoted $\widehat{\mathcal{C}}$. Specifically, here we use an algorithm that, on input a circuit in $\widehat{\mathcal{C}}$, distinguishes the case that circuit evaluates to 1 on all inputs and the case that circuit evaluates to 1 on at most half of the inputs. Assuming that (n-bit input) circuits in $\mathcal{C}$ can be described by strings of length at most $2^{n/4}$, the (lower bound) proof uses the following facts:

1. A hierarchy theorem for non-deterministic time as in Take 1; specifically, the fact that $\mathrm{NTime}(2^n/n^{\omega(1)})$ is strictly contained in $\mathrm{NTime}(2^n)$.

   Again, assuming towards the contradiction that $\mathcal{X} \subseteq \mathcal{C}$, we shall derive a contradiction by starting with any $S \in \mathrm{NTime}(2^n)$, and showing that $S \in \mathrm{NTime}(2^n/n^{\omega(1)})$.

[5]Here we rely on the hypothesis that the m-bit input circuits in $\mathcal{C}$ can be described by $2^{m/4}$ bits (and on $n' = n + O(\log n) < 2n$).

2. The main deviation from Take 1 takes place here. Rather than using the standard non-deterministic machine for $S \in \mathrm{NTime}(2^n)$, we use a PCP system for this set. Specifically, on input x, this PCP uses proofs of length $N' \stackrel{\mathrm{def}}{=} \widetilde{O}(2^{|x|})$, and its verifier has randomness complexity $n' = \log_2 N' = |x| + O(\log |x|)$ and (non-adaptive) query complexity $q \stackrel{\mathrm{def}}{=} \mathrm{poly}(|x|)$. Furthermore, on input x, the verifier's (non-adaptive) queries and its final decision can be implemented by a (polynomail-time constructable) $\mathcal{AC}_0$-type circuit, denoted V_x. The standard completeness and soundness conditions (of PCPs) assert:

   - If $x \in S$, then there exists $\pi \in \{0,1\}^{N'}$ such that $\Pr_{r \in \{0,1\}^{n'}}[V_x^w(r)=1] = 1$.
   - If $x \notin S$, then for every $\pi \in \{0,1\}^{N'}$ it holds that $\Pr_{r \in \{0,1\}^{n'}}[V_x^w(r)=1] \leq 1/2$.

   Needless to say, establishing the existence of the foregoing PCP system is highly non-trivial. I view that result as part of the third generation of PCP constructions. (Note that here the proof length is almost linear in the length of the standard (NP-type) witness, whereas the standard PCP construction (of the first generation) yields proof of length that is a (large) polynomial in the witness length.)

3. The fact that $\mathcal{X}$ contains a function $\Pi : \{0,1\}^* \times \mathbb{N}' \to \{0,1\}$ such that for every $x \in S$ the function $\Pi_x : [N'] \to \{0,1\}$ defined as $\Pi_x(i) \stackrel{\mathrm{def}}{=} \Pi(x,i)$ (for every $i \in [N']$) represents a PCP-oracle that makes V_x always accept (i.e., $\Pr_{r \in \{0,1\}^{n'}}[V_x^{\Pi_x}(r)=1] = 1$).[6]

4. Next, assuming towards the contradiction that $\mathcal{X} \subseteq \mathcal{C}$, it follows that the function Π can be computed by a family of circuits, denoted $\{C_m\}_{m \in \mathbb{N}}$, that is in $\mathcal{C}$. For every $x \in \{0,1\}^n$, we let $C'_x(i) = C_{n+n'}(x,i)$, for every $i \in [N']$.

   Recall that if $x \in S$, then $\Pr_{r \in \{0,1\}^{n'}}[V_x^{C'_x}(r)=1] = 1$, and otherwise (i.e., $x \notin S$) it holds that $\Pr_{r \in \{0,1\}^{n'}}[V_x^{C'}(r)=1] \leq 1/2$ for every C. A key observation is that for a fixed value of r, the circuit V_x invokes C'_x on $\mathrm{poly}(|x|)$ values (which are determined by x and r). Hence, the final decision of V_x on input r (when given oracle access to C'_x) can be represented by the circuit $\widehat{c}_x(r) = D_x(r, C'_x(Q_1(x,r)), ..., C'_x(Q_q(x,r)))$, where Q_i denotes computation of the i^{th} query and D_x denotes the computation of the final decision.

5. Lastly, assuming that we can distinguish in time $2^{|x|}/|x|^{\omega(1)}$ between the case that $\Pr_r[\widehat{c}_x(r)=1] = 1$ and the case that $\Pr_r[\widehat{c}_x(r)=1] \leq 1/2$, we conclude that S is in $\mathrm{NTime}(2^n/n^{\omega(1)})$, by presenting a non-deterministic machine that on input $x \in \{0,1\}^n$ first guesses $C_{n+n'}$, by making $2^{(n+n')/4}$ non-deterministic steps, then constructs $\widehat{c}_x$, and finally invokes the foregoing decision procedure on $\widehat{c}_x$. Note that the circuit $\widehat{c}_x$ is obtained by composing q copies of $C'_x = C_{n+n'}(x,\cdot)$ with circuits that compute the Q_i's and D_x. Furthermore, these ($\mathcal{AC}_0$-type) circuits can be constructed in polynomial-time.

Indeed, for the foregoing to yield $S \in \mathrm{NTime}(2^n/n^{\omega(1)})$, we need an algorithm as postulated in the last item. Specifically, defining $\widehat{\mathcal{C}}$ as the class of circuits obtained by composition of circuits in $\mathcal{C}$

[6] For $\mathcal{X} = \mathcal{E}^{\mathcal{NP}}$ an adequate function Π_x can be computed by determining its values sequentially. Specifically, $\Pi_x(i)$ is set to 0 if and only if the i-bit long string $\pi' \stackrel{\mathrm{def}}{=} \Pi_x(1) \cdots \Pi_x(i-1) \cdot 0$ is a prefix of an N'-bit long string that makes V_x accept with probability 1 (i.e., $\exists \pi''$ such that $\Pr_{r \in \{0,1\}^{n'}}[V_x^{\pi' \pi''}(r)=1] = 1$). Again, the relevant NP-query is $(x, \pi' 10^{N'-i})$.

and $\mathcal{AC}_0$ as in the last item, we need an $2^n/n^{\omega(1)}$-time algorithm that given an n'-bit circuit in $\widehat{\mathcal{C}}$ distinguishes circuits that evaluate to 1 with probability 1 from circuits that evaluate to 1 with probability at most 1/2.

The missing link. Indeed, in both takes, we avoided the chore of designing suitable algorithms. We stress that this is known for the case that $\widehat{\mathcal{C}} = \mathcal{ACC}$, which means that non-trivial derandomization is known for $\mathcal{ACC}$.

Digest. The method, as described above, relies on guessing a circuit (in the class $\mathcal{C}$); hence, the resulting decision procedure for S is non-deterministic, whereas its running time is dominated by the running time of the circuit-analysis procedure for $\widehat{\mathcal{C}}$ (which is larger that the size of the guessed circuit). Contradiction to $\mathcal{X} \subseteq \mathcal{C}$ requires a circuit-analysis procedure for $\widehat{\mathcal{C}} \supseteq \mathcal{C}$ that is mildly more efficient than the non-deterministic time complexity of S. (In Take 1, this was a procedure for deciding whether a given circuit is identical to 1, and in Take 2 it was a procedure for approximating the probability that the circuit evaluates to 1.) It follows that *if circuits in a class that slightly extends $\mathcal{C}$ can be analyzed within complexity that is mildly better than the straightforward algorithm, then $\mathcal{C}$ cannot solve all problems in $\mathcal{X}$.*

Acknowledgements. These notes are based on Roei Tell's exposition *Non-trivial derandomization implies weak lower bounds: An (almost) elementary proof.*[7]

[7] Available from `https://sites.google.com/site/roeitell/Expositions`.