Nonuniform ACC Circuit Lower Bounds: A Summary

Sai Bavisetty (vb8), Rucha Kulkarni (ruchark2)

December 8, 2021

Abstract

This article is a synopsis of Williams' seminal paper titled *Nonuniform* ACC *circuit lower bounds[Wil14]*. The author proves two nonuniform (circuit) lower bound results for certain uniform time complexity classes. First, it is shown that the class NEXP, in fact $\mathsf{NTIME}[2^n]$, does not have polynomial sized ACC circuits. The second result proves that E^{NP} does not have nonuniform ACC circuits of sub-exponential size. The following size-depth trade-off result is implied by the later result. For every d, m, there is a $\delta > 0$ such that E^{NP} does not have depth-d ACC circuits with MOD_m gates of 2^{n^δ} size.

The key idea to prove these results is to prove by contradiction, that the existence of such small circuits for the respective classes, coupled with fast algorithms (of a specific worst case time bound) for deciding $\mathsf{ACC} - \mathsf{SAT}$, which are SAT formulae based on ACC circuits, together imply $\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[2^n/n^k]$, for some integer k. This violates the non-deterministic time hierarchy theorem. Such fast algorithms for $\mathsf{ACC} - \mathsf{SAT}$ are provided, implying the results.

1 Introduction

Computational Complexity traditionally used uniform models of computation to establish the worst case behaviour of a problem. Shannon introduced the concept of circuit complexity, by proving the existence of a boolean function on n variables that requires an $\Omega(2^n/n)$ sized circuit. Nonuniform computation gained popularity and today we have a rich literature for worst case nonuniform complexity analysis. A central question in complexity theory today is to relate nonuniform computation with traditional (uniform) computation. A more specific question of particular interest is if this theory helps in establishing impossibility results for uniform complexity classes. For example, $\mathsf{P} \neq \mathsf{NP}$ is implied if we can prove that there is a problem in NP that cannot be solved by any circuit family where the n^{th} circuit has size at most polynomial in n.

The following approach to prove $\mathsf{P} \neq \mathsf{NP}$ arises from this direction of thought. Start with a very restricted nonuniform computational model. Then find a

function in NP that requires circuits larger than available in this restricted class. Gradually lift these restrictions, proving such lower bounds for larger classes, finally establishing super-polynomial size circuit lower bounds for NP.

Working along these lines led to several results. First, the PARITY function of n bits was proven to not belong in polynomial-sized AC, the class of constant depth circuits made from AND, OR and NOT gates of arbitrary fan-in [FSS84, Ajt]. This lower bound was improved [Yao85], and proved tight [Has86], to show PARITY $\notin$ exponential-sized AC. Then the CLIQUE problem, known to be NP $-$ Complete, was shown to require super-polynomial monotone circuits [Raz85]. A later result improved this bound to exponential sized monotone circuits [AB87]. However, Razborov [Raz89] proved that the techniques used in proving the results for CLIQUE would probably not extend to general circuits.

Lifting the restrictions on circuits slightly, results were proven for the class of AC circuits that also additionally have $\mathsf{MOD_m}$ gates for a fixed m. The MAJORITY function of n bits was proved to require exponential sized circuits of this type for $m = 2$ [Raz87]. The $\mathsf{MOD_q}$ function for a prime number q was proved to require exponential sized such circuits when m is a fixed prime number distinct from q [Smo87]. Study of a further generalized class of circuits, denoted ACC, that allows $\mathsf{MOD_m}$ gates for any $m > 1$, was then suggested by [Bar89], but results here remained elusive.

After this initial success, no further results were obtained for a few years. Instead of trying to answer "which simple (NP) function cannot be computed with weak circuits", one could then attempt to see if such an answer can be found for complicated functions. Stated differently, understanding the bounds of a nonuniform class in terms of a uniform class, that is, finding the largest uniform complexity class that can be expressed using weak circuits, could be fruitful. This is the broad question that motivated this paper.

The question then remains is the choice of the uniform and the nonuniform class to relate in this way, so that the relations have interesting implications. As progress along the prior approach of first choosing the nonuniform class and proving its limitations had halted at ACC, this was a natural choice of nonuniform computation. A series of results indicated that NEXP would be a uniform class for which proving ACC lower bounds would have non-trivial consequences [BFNW91, IKW02, KVM02, NW94]. For instance, this would imply that Merlin-Arthur games can be simulated by nondeterministic algorithms.

The following questions thus became frontier questions in complexity theory, and are the focus of this paper.

1. Does non-deterministic $2^{O(n)}$ time have nonuniform polynomial sized ACC circuits? That is, is $\mathsf{NTIME}[2^{O(n)}] \subseteq$ nonuniform ACC?
2. Is $\mathsf{E^{NP}} \subseteq \mathsf{P/poly}$?

1.1 Contribution

The first question was resolved, with a negative answer. The second was partially resolved by proving ACC (instead of P/poly) lower bounds for $\mathsf{E^{NP}}$. For-

mally, the two main results of this paper are as follows.

Theorem 1.1. $\mathsf{NTIME}[2^{\mathsf{O}(\mathsf{n})}]$ *does not have nonuniform* ACC *circuits of polynomial size.*

Theorem 1.2. *[Exponential Size-Depth Trade-off] For every* d *and* m*, there is a* $\delta > 0$ *and a language in* E^{NP} *that fails to have nonuniform* ACC *circuits with* $\mathsf{MOD}_{\mathsf{m}}$ *gates of depth* d *and size* $2^{n^{\delta}}$*.*

2 Preliminaries

We assume familiarity with basic concepts in computational complexity like the classes NEXP and $\mathsf{NTIME}[\mathsf{t}(\mathsf{n})]$, for some time bound $t(n)$. For a refresher, we refer to the book Arora et al [AB09]. We state the definitions of the primary complexity classes and the problems centrally used in the paper.

Definition 2.1 (ACC)**.** *The class* ACC *consists of constant-depth circuit families over the basis* AND*,* OR*,* NOT*, and* $\mathsf{MOD}_{\mathsf{m}}$ *for arbitrary constant* $m > 1$*.*

$\mathsf{ACC} - \mathsf{SAT}$ and $\mathsf{ACC} - \mathsf{CIRCUIT} - \mathsf{SAT}$ refer to the SAT and $\mathsf{CIRCUIT} - \mathsf{SAT}$ problems defined on ACC circuits.

Definition 2.2 ($\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$)**.** *For a convenient encoding for circuits and boolean formulae, denote by* ϕ_C *the string obtained by evaluating a circuit* C *on all it's inputs. The language* $\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ *is then defined as the set of all circuits whose corresponding formula* ϕ_C *is satisfiable.*

$$\mathsf{SUCCINCT} - 3 - \mathsf{SAT} = \{\textit{Circuits } C : \phi_{\mathsf{C}} \in \mathsf{SAT}\}.$$

$\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ is a known $\mathsf{NEXP} - \mathsf{Complete}$ problem.

Definition 2.3 (E^{NP})**.** E^{NP} *is the class of languages that can be decided in time* $\mathsf{NTIME}[2^{\mathsf{O}(\mathsf{n})}]$ *with an* NP *oracle.*

Definition 2.4 (SYM^{+})**.** *This is the class of depth-2 boolean circuits that compute a symmetric function (at the output gate) of the* AND *of input variables (at the second layer).*

3 Technical Overview

The proof of Theorem 1.2 can be broadly divided into two main steps, that intuitively prove the following statements.

1. $\mathit{Fast} - \mathsf{ACC} - \mathsf{SAT}$ AND $\mathsf{E}^{\mathsf{NP}} \subseteq \mathit{Small} - \mathsf{ACC} \Rightarrow \mathsf{NTIME}[2^{\mathsf{O}(\mathsf{n})}] \subseteq \mathsf{NTIME}[2^{\mathsf{O}(\mathsf{n})}/\mathsf{n}^{\mathsf{k}}]$. That is, fast algorithms for $\mathsf{ACC} - \mathsf{SAT}$ and Small ACC circuits for E^{NP} cannot both exist, else the non-deterministic time hierarchy theorem gets violated. The terms "fast" and "small" are suitably formalized in Section 3.1.

2. $\mathsf{ACC} - \mathsf{SAT}$ is fast. That is, a fast algorithm for $\mathsf{ACC} - \mathsf{SAT}$ exists, implying the second assumption in the previous statement must be false.

Proving the first statement uses two facts. First, every language in $L \in \mathsf{NTIME}[2^n]$ can be reduced to a circuit $C \in \mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ [1]. This follows by applying a padding argument to a similar reduction shown for languages in $\mathsf{NTIME}[n]$ to SAT [FLVMV05, Tou01]. Second, if $\mathsf{E}^{\mathsf{NP}} \subseteq 2^{n/4}$-sized ACC, then for every language in $\mathsf{NTIME}[2^n]$, there is an $2^{3n/4}$-sized ACC circuit that encodes a satisfying assignment for the SAT formula ϕ_C. This can be proven by giving a E^{NP} algorithm which gives the satisfying assignment, implying the existence of an ACC circuit, that encodes a satisfying assignment of ϕ_C.

In short, L can be represented by ϕ_C, an exponential sized formula, whose satisfying assignment can be encoded by a circuit. This then implies that if we could decode this circuit non-deterministically (verify this circuit exists) in time $\mathsf{NTIME}[2^n/n^k]$, then every such language L can be solved in sub-exponential time. But $L \in \mathsf{NTIME}[2^n] \Rightarrow \mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[2^n/n^k]$, a contradiction.

However, they show such a "fast" (non-deterministic sub-exponential time) algorithm only for ACC circuits, hence it is essential that the reductions generate an ACC circuit. This is shown via a clever construction reducing the possibly non-ACC circuit C to an ACC circuit C^{ACC}, described later.

A proof of the second statement, that is a "fast" algorithm for ACC circuits, is the last element, establishing the result. An equivalent SYM^+ circuit is shown for every ACC circuit. An algorithm to rapidly evaluate this circuit on all inputs is then described. There are multiple algorithms for this step, perhaps the easiest is one using a dynamic programming technique. We describe the reduction and the algorithm in Section 3.2.

Theorem 1.1 is proved using the same ingredients and approach. The difference is in the statement of fact 2, which proved a succinct satisfying assignment for $\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ exists if E^{NP} has small circuits. Instead, we must now prove a similar statement assuming small circuits for $\mathsf{NTIME}[2^n]$. To prove this, earlier an E^{NP} machine was demonstrated to construct the succinct SAT assignment. Now a simple reduction from an $\mathsf{NTIME}[2^n]$ language to a canonical $\mathsf{NEXP} - \mathsf{Complete}$ problem is shown. Small circuits for $\mathsf{NTIME}[2^n]$ then imply small circuits for all of NEXP, thus for $\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ too. The rest of the proof then follows in the same way.

3.1 $Fast-\mathsf{ACC} - \mathsf{SAT}$ AND $\mathsf{E}^{\mathsf{NP}} \subseteq Small-\mathsf{ACC} \Rightarrow \mathsf{NTIME}[2^{O(n)}] \subseteq \mathsf{NTIME}[2^{O(n)}/n^k]$

Theorem 3.1 (Reduction to $\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$)**.** *There is a constant $c > 0$ such that for every $L \in \mathsf{NTIME}[2^n]$, there is a reduction from L to $\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ which on input x of length n runs in $poly(n)$ time and produces a circuit C_x with at most $n + clogn$ inputs and $O(n^c)$ size, such that $x \in L$ if and only if the decompressed formula ϕ_{C_x} of $2^n poly(n)$ size is satisfiable.*

[1] We use L as notation for an instance of $\mathsf{NTIME}[2^n]$, C as the corresponding reduced $\mathsf{SUCCINCT} - 3 - \mathsf{SAT}$ instance, and ϕ_C for the formula encoded by C hence forth

It turns out that, if we assume E^{NP} has small circuits then in fact the $\mathsf{SUCCINCT}-3-\mathsf{SAT}$ instances actually have a succinct satisfying assignment.

Theorem 3.2 (Succinct satisfying assignment)**.** *If E^{NP} has ACC circuits of size $2^{\frac{n}{4}}$, then there is a fixed constant c such that for every language $L \in \mathsf{NTIME}[2^n]$ and every $x \in L$ of length n, there is a circuit W_x of size at most $2^{\frac{3n}{4}}$ with $k \leq n + c\log n$ inputs such that the variable assignment $z_i = W(i)$ for all $i = 1, ..., 2^k$ is a satisfying assignment for the formula ϕ_{C_x} , where C_x is the circuit obtained by the reduction in Theorem 3.1.*

Using Theorem 3.1 and Theorem 3.2, we have $x \in L \iff C_x \in \mathsf{SUCCINCT}-3-\mathsf{SAT} \iff \exists \mathsf{W}$ such that $\mathsf{z_i} = \mathsf{W(i)}$ is a satisfying assignment
Since we have faster algorithm for satisfiability we would like to have an equivalence with satisfiability of a circuit.

3.1.1 Formulation in terms of satisfiability

We consider the ciruit D_x, which consists of a circuit C' which takes i as input and outputs the the the indices of the variables in the i clause along with the sign bits. The indices of the variables are then input into W, which outputs the assignment for the three variables. D_x then checks if this satisfies the i clause. If it is satified then it outputs 0. So D_x is unsatisfiable if and only if $\exists W$ which encodes a satisfying assignment for ϕ_{C_x}.
We now have that $x \in L \iff D_x$ is unsatisfiable. If D_x were an ACC circuit, we can show that any $L \in \mathsf{NTIME}[2^n]$ can be recognised in $\mathsf{NTIME}[o(2^n)]$.

3.1.2 Algorithm

Given a string x of length n, compute the $\mathsf{SUCCINCT}-3-\mathsf{SAT}$ circuit C_x in polynomial time and nondeterministically guess a $2^{\frac{3n}{4}}$ size circuit W. Now the goal is to check that W succinctly encodes a satisfying assignment for the underlying formula ϕ_{C_x}. For this consider the circuit D_x and by running a fast enough $\mathsf{ACC}-\mathsf{SAT}$ algorithm we can do this in $\mathsf{NTIME}[o(2^n)]$. To see that this algorithm is correct, observe there is a size-$2^{\frac{3n}{4}}$ circuit W such that D is an unsatisfiable circuit, if and only if there is such a W encoding a satisfying assignment for ϕ_{C_x} , if and only if $x \in L$.

But unfortunately the reduction to $\mathsf{SUCCINCT}-3-\mathsf{SAT}$ does not usually give an ACC circuit. So the idea is to replace it by an equivalent ACC circuit.

Lemma 3.1 (Equivalent ACC circuit)**.** *There is a fixed $d > 0$ with the following property. Assume P has ACC circuits of depth d' and size at most $2^{\frac{n}{4}}$. Further assume ACC CIRCUIT SAT on circuits with $n + c\log n$ inputs, depth $2d(3n) + b$, and at most $O(2^{3n/4} + n2^{n/2})$ size can be solved in $O(2^n/n^c)$ time, for sufficiently large $c > 2d$. Then for every $L \in \mathsf{NTIME}[2^n]$, there is a nondeterministic algorithm A such that:*

1. *A runs in* $O(2^n/n^c + 2^{3n/4}poly(n))$ *time*
2. *for every* x *of length* n, $A(x)$ *either prints reject or it prints an* ACC *circuit* C_x^{ACC} *with* $n + dlogn$ *inputs, depth* d', *and* $2^{\frac{n+dlogn}{4}}$ *size, such that* $x \in L$ *if and only if* C_x^{ACC} *is the compression of a satisfiable* $3-CNF$ *formula of* $2^n.poly(n)$ *size*
3. *there is always at least one computation path of* $A(x)$ *that prints the circuit* C_x^{ACC}

3.2 $\mathsf{ACC} - \mathsf{SAT}$ is Fast

The main idea for this fast algorithm is to first produce an equivalent SYM^+ circuit for the given ACC circuit.

Lemma 3.2 (Equivalent SYM circuit)**.** *There is an algorithm and function* $f : \mathbb{N} \to \mathbb{N}$ *such that given an* ACC *circuit of depth* d *and size* s, *the algorithm outputs an equivalent* SYM^+ *circuit of* $s^{O(log^{f(d,m)}s)}$ *size. The algorithm takes at most* $s^{O(log^{f(d,m)}s)}$ *time. Furthermore, given the number of* AND *in the circuit that evaluate to 1, the symmetric function itself can be evaluated in* $s^{O(log^{f(d,m)}s)}$ *time.*

Theorem 3.3. *Suppose the size* $s < 2^{0.1n}$, *then the function which calculates the number of* AND *gates that evaluate to* 1, *can be computed in* $O(2^n poly(n))$, *where* n *is the number of inputs.*

Proof. Suppose the variables are $x_1, \ldots, x_n$. Let $V = \{x_1, \ldots, x_n\}$. For every subset $S \subset V$, define χ_S to be the input

$$x_i = \begin{cases} 1 & \text{if } x_i \in S \\ 0 & \text{otherwise} \end{cases}$$

Let $g(S)$ be the function we want to compute then,

$$g(S) = \{\text{Number of AND gates that evaluate to 1 when the input is } \chi_S\}$$

To compute g, we start out with another function f defined to be

$$f(S) = \{\text{Number of AND gates whose input variables are exactly S}\}$$

f can be computed by just creating a lookup table of size 2^n and for each of the AND gates increment the corresponding cell in the table by 1. This takes $O(2^n + s.poly(n)) = O(2^n)$ time. Observe that $g(S) = \Sigma_{T \subset S} f(T)$. We calculate g inductively from f by defining

$$g_i(T) = \begin{cases} g_{i-1}(T) + g_{i-1}(T \backslash \{i\}) & \text{if } i \in T \\ g_{i-1}(T) & \text{otherwise} \end{cases}$$

It follows that g_{i+1} can be computed from g_i in $O(2^n poly(n))$ and $g_0 = f$ and $g_n = g$. This proves that we can compute g given f in $O(2^n poly(n))$ time. □

Theorem 3.4 (Fast algorithm for ACC-SAT)**.** *For every $d > 1$ there is an $\epsilon \in (0,1)$ such that satisfiability of depth-d ACC circuits with n inputs and 2^{n^ϵ} size can be determined in $2^{n-\Omega(n^\delta)}$ time for some $\delta > \epsilon$ that depends only on d.*

Proof. Given a depth d, ACC circuit of size 2^{n^ϵ} and n inputs we consider a circuit C' with $n-l$ inputs and size $2^l.2^{n^\epsilon}$ which is obtained by taking 2^l copies of C and plugging in different possible inputs for l inputs into each copy and then taking an OR of all the values.
We use Lemma 3.2 to get the equivalent SYM^+ circuit C'' whose size is $s' = 2^{(n^\epsilon+l)O(log^{f(d,m)}2^{n^\epsilon+l})} = 2^{O((n^\epsilon+l)^{f(d,m)+1})}$ Choosing $l = n^{\frac{1}{2f(d,m)}}$ and $\epsilon \leq \frac{1}{2f(d,m)}$ we get $s \leq 2^{n^{2/3}}$.
By Lemma 3.2 we know we can evaluate this circuit in $2^{O(n^{2/3})}$ once we know the number of AND gates that evaluate to 1. By Theorem 3.3, the function which evaluates the number of AND gates takes $O(2^{n-l}poly(n)) \leq 2^{n-\Omega(n^{\frac{1}{2f(d,m)}})}$. □

3.3 Proof of ACC lower bound for E^{NP}

Proof. Given $x \in L$ of size n, by Lemma 3.1, we get an $\mathsf{ACC}-\mathsf{SAT}$ instance of size $O(2^{3n/4})$ and $n+clogn$ inputs in $O(\frac{2^n}{n^c})$. By Theorem 3.4, this satisfiability instance can be solved in $2^{n+clogn-\Omega(n+clogn)^\delta} \leq O(\frac{2^n}{n^c})$. So the total time is $O(\frac{2^n}{n^c})$, which is a contradiction to the Time Hierarchy theorem. □

4 Other details

There are a few subtle details worth pointing out. A detailed discussion on each of these is interesting but beyond the scope of this article, and can be found in the original work [Wil14].

- The $\mathsf{NTIME}[2^n]$ bound of Theorem 1.1 is actually stronger. The reduction from $\mathsf{NTIME}[2^n]$ to NEXP generates NEXP instances of a particular size, which when applied without approximating to 'poly-size' implies ACC circuits of 'half-exponential type function'-sizes, not just polynomial sizes, cannot exist.

- The results of the paper apply to circuit classes more general than ACC, for any class that (a) contains AC, and (b) is closed under composition.

- There are several known obstacles in lower bound proving techniques, like the natural proofs barrier [RR97], and relativization [BGS75] and algebraization [AW09]. These are successfully bypassed by the techniques used in this paper.

- The results involving uniform classes like $\mathsf{E}^{\mathsf{NP}}, \mathsf{NTIME}[t(n)]$ assume the corresponding languages run on multi-tape turing machines. The relevant

alternate automata (RAMs) are known to have equivalent (within asymptotic limits) running times [GS89], hence the parameters computed, and all results hold true for such alternate uniform machine models too.

References

[AB87] Noga Alon and Ravi B Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[Ajt] M Ajtai. $\sum_1^1$ -formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1).

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory (TOCT)*, 1(1):2, 2009.

[Bar89] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BFNW91] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. Bpp has subexponential time simulations unless exptime has publishable proofs. In *[1991] Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 213–219. IEEE, 1991.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the p=?np question. *SIAM Journal on computing*, 4(4):431–442, 1975.

[FLVMV05] Lance Fortnow, Richard Lipton, Dieter Van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM (JACM)*, 52(6):835–865, 2005.

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984.

[GS89] Yuri Gurevich and Saharon Shelah. Nearly linear time. In *International Symposium on Logical Foundations of Computer Science*, pages 108–118. Springer, 1989.

[Has86] John Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. Citeseer, 1986.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[KVM02] Adam R Klivans and Dieter Van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

[Raz85] Alexander A Razborov. Lower bounds for the monotone complexity of some boolean functions. In *Soviet Math. Dokl.*, volume 31, pages 354–357, 1985.

[Raz87] Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

[Raz89] Alexander A Razborov. On the method of approximations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 167–176. ACM, 1989.

[RR97] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

[Tou01] Iannis Tourlakis. Time–space tradeoffs for sat on nonuniform machines. *Journal of Computer and System Sciences*, 63(2):268–287, 2001.

[Wil14] Ryan Williams. Nonuniform acc circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 1–10. IEEE, 1985.