The Composition Complexity of Majority

Victor Lecomte
Stanford University, CA, USA

Prasanna Ramakrishnan
Stanford University, CA, USA

Li-Yang Tan
Stanford University, CA, USA

───── **Abstract** ─────

We study the complexity of computing majority as a composition of *local* functions:

$$\text{MAJ}_n = h(g_1, \ldots, g_m),$$

where each $g_j : \{0,1\}^n \to \{0,1\}$ is an arbitrary function that queries only $k \ll n$ variables and $h : \{0,1\}^m \to \{0,1\}$ is an arbitrary combining function. We prove an optimal lower bound of

$$m \geq \Omega\left(\frac{n}{k}\log k\right)$$

on the number of functions needed, which is a factor $\Omega(\log k)$ larger than the ideal $m = n/k$. We call this factor the *composition overhead*; previously, no superconstant lower bounds on it were known for majority.

Our lower bound recovers, as a corollary and via an entirely different proof, the best known lower bound for bounded-width branching programs for majority (Alon and Maass '86, Babai et al. '90). It is also the first step in a plan that we propose for breaking a longstanding barrier in lower bounds for small-depth boolean circuits.

Novel aspects of our proof include sharp bounds on the information lost as computation flows through the inner functions g_j, and the bootstrapping of lower bounds for a multi-output function (Hamming weight) into lower bounds for a single-output one (majority).

1 Introduction

A basic theme in computer science is the representation of certain functions as the combination of simpler ones. Indeed, the field of distributed computing and the widespread principle of divide-and-conquer rely on this property of functions.

In this paper we focus on *locality* as our notion of simplicity: a k-local function over n variables is one that depends only on $k \ll n$ input coordinates. This leads us to the following complexity measure of boolean functions, first studied by Hrubeš and Rao [15], which quantifies how easily they can be represented as the combination of local functions:

▶ **Definition 1** (composition complexity). *The k-composition complexity of a function f, denoted $\mathsf{CC}_k(f)$, is the minimum m such that f can be expressed as $h(g_1, \ldots, g_m)$, where each of the inner functions g_j queries only k variables.*

Clearly, any function f that actually depends on all n variables must have $\mathsf{CC}_k(f) \geq n/k$, since every variable must be queried at least once. The parity function shows that this bound can be tight: $\mathsf{CC}_k(\text{PARITY}_n) \leq O(n/k)$, since we can let each g_j compute the parity of a set of k variables, and let h compute the parity of these parities. However, some functions inherently require more than n/k inner functions: they incur a *composition overhead*. This motivates defining the *k-composition overhead* of f to be the ratio $\frac{\mathsf{CC}_k(f)}{n/k}$ between its k-composition complexity and the ideal n/k.

1.1   This work

In this paper we study the composition complexity of the majority function. Prior to our work, this was perhaps the most basic function whose composition complexity was not well understood. For an upper bound, it is not hard to see that $\mathsf{CC}_k(\text{MAJ}_n) \leq O(\frac{n}{k} \log k)$. We can split the variables into n/k disjoint sets of size k and devote $O(\log k)$ of the inner functions g_i to computing the Hamming weight of each set. Then h can determine the overall Hamming weight, and output 1 if and only if it is at least $n/2$. This uses a total of $m \leq O(\frac{n}{k} \log k)$ inner functions.

As for lower bounds, while it is an easy exercise to improve the trivial lower bound of $\geq n/k$ to $> n/k$ in the case of majority, even a modest lower bound of $\geq 1.1\frac{n}{k}$ seems challenging to establish. Our main result is an asymptotically tight lower bound showing that the construction described above is optimal, and that majority has a composition overhead of $\Theta(\log k)$.

▶ **Theorem 2** (Main theorem). *For all $\epsilon > 0$ and $k \leq n^{1-\epsilon}$, $\mathsf{CC}_k(\text{MAJ}_n) \geq \Omega(\frac{n}{k} \log k)$.*

In addition to being a natural complexity measure that is of independent interest, our study of composition complexity is further motivated by its relationships to two important models of computation: bounded-width branching programs and small-depth circuits. These are two of the most intensively studied models in circuit complexity, and majority plays a starring role in the efforts at lower bounds for both models. As we now elaborate, Theorem 2 recovers, as a corollary and via an entirely different proof, the current best lower bounds on the length of bounded-width branching programs for majority [2, 4]. It is also the first step in a plan that we propose for lower bounds against depth-3 circuits computing majority, a long-standing open problem that dates back to the 1990s [13]; such lower bounds would represent the first improvements over the state of the art for depth-3 circuits in over three decades [12].

1.2   Motivation and implications

1.2.1   Bounded-width branching programs

There is an easy reduction from branching programs to our model: if function f is computed by a bounded-width branching program of length L, then $\mathsf{CC}_k(f) \leq O(L/k)$. The reduction works by cutting the branching programs into $\sim L/k$ segments of length at most k, then for each segment and for each state s at the start of the segment, use $O(1)$ inner functions to compute which state one would end up at the end of the segment if one started from s. Since the segments have length at most k, each of the inner functions depends on at most k variables.

This means that nontrivial lower bounds on the composition complexity $\mathsf{CC}_k(f)$ for any k directly imply nontrivial lower bounds on the length of bounded-width branching programs: $L \geq \Omega(k \cdot \mathsf{CC}_k(f))$. In particular, setting $k = \sqrt{n}$, by Theorem 2 we obtain

that bounded-width branching programs computing the majority function must have length $L \geq \Omega(\sqrt{n} \cdot \frac{n}{\sqrt{n}} \log \sqrt{n}) = \Omega(n \log n)$, recovering the classic lower bound of Alon and Maass [2] and Babai, Pudlák, Rödl, and Szemerédi [4] as a corollary:

▶ **Theorem 3** ([2, 4]). *Any bounded-width branching program computing* MAJ_n *must have length* $\Omega(n \log n)$.

This lower bound remains the current state of the art. Interestingly, our techniques are completely different from the techniques used in [2] and [4]: they first prove a Ramsey-theoretic lemma that identifies two sets of variables that are queried in disjoint segments of the branching program, then conclude by a communication complexity argument between them. We elaborate on our techniques in Section 1.3.

1.2.2 Lower bounds for small-depth circuits computing majority

We view Theorem 2 in part as an essential and necessary first step towards proving stronger lower bounds against small-depth circuits. In this section, we will outline why such lower bounds are particularly interesting, and the relationship between our composition complexity lower bounds and circuit lower bounds.

Lower bounds for small-depth circuits

A fruitful line of work from the '80s [7, 1, 32, 12] managed to prove strong lower bounds against small-depth circuits, but using a surprisingly simple function: parity. In particular, the result that culminated from these works was that any depth-d circuit computing PARITY_n must have size $2^{\Omega\left(n^{\frac{1}{d-1}}\right)}$, which is superpolynomial for any $d = o(\log n / \log \log n)$. These bounds are optimal for parity – an extension of the divide-and-conquer scheme mentioned earlier matches this bound. As we now elaborate, these remain essentially the strongest (explicit) lower bounds against small-depth circuits we have for any function, even in the case of $d = 3$, despite the fact that counting arguments give lower bounds of $\Omega(2^n/n)$, and we expect that circuits for hard functions like SAT must also have size $2^{\Omega(n)}$.

Depth-3 circuits and majority

The problem of improving the state of the art for depth-3 circuits ($2^{\Omega(\sqrt{n})}$ for PARITY_n) in particular has received significant attention as one of the simplest restricted models that are poorly understood. Stronger depth-3 bounds are likely to imply stronger small-depth bounds in general, and *much* stronger bounds of the form $2^{\omega(n/\log \log n)}$ would also give functions that cannot be computed by linear-size log-depth circuits due to a classical result of Valiant [30]. For a detailed exposition of the motivations for depth-3 and the attempts to understand the model, see [16, Chapter 11].

Of the functions that could prove stronger depth-3 lower bounds, majority, being such a basic and simple-to-understand function, is a particularly enticing candidate. The divide-and conquer construction analogous to that for parity gives depth-3 circuits of size $2^{O(\sqrt{n \log n})}$, and the same $2^{\Omega(\sqrt{n})}$ lower bound for parity also applies to majority.[1] In light of this gap, Håstad, Jukna, and Pudlák [13] posed the following challenge.

---

[1] For general depth d, the upper bound is $2^{O\left(n^{\frac{1}{d-1}} \cdot (\log n)^{1 - \frac{1}{d-1}}\right)}$ [17] and the lower bound is $2^{\Omega\left(n^{\frac{1}{d-1}}\right)}$ once again.

▶ **Open Problem 1** ([13]). *Find an explicit function that requires depth-3 circuits of size* $2^{\omega(\sqrt{n})}$. *In particular, does* MAJ_n *require depth-3 circuits of that size?*

Despite this natural candidate function, all of the improvements on the lower bounds for depth-3 circuits have still been of the form $2^{c\sqrt{n}}$ for successively larger c. [13] found an innovative method of proving lower bounds for circuits from the "top down", and were able to get constants $c = 0.618$ and 0.849 for parity and majority respectively. Paturi, Pudlák, and Zane [24] improved the constant to the optimal $c = 1$ for parity, and later the same authors along with Saks [23] obtained the constant $c = 1.282$ for the membership function of an error-correcting code. Getting a super-constant improvement over this state of the art has been a major frontier of circuit complexity for decades.

For majority, there have also been attempts at improved *upper bounds*. [31] proposed a probabilistic construction of a depth-3 circuit computing MAJ_n with size $2^{O(\sqrt{n})}$, but the construction turned out to be mistaken.[2]

The need for new techniques

One view of why previous techniques have fallen short on resolving Open Problem 1 is that at their core, they use sensitivity as the key complexity measure for which small depth circuits are weak (see [5, 28, 20]). With respect to sensitivity, of course PARITY_n is the most complex function because it has sensitivity n at every input. The fact that parity is the hardest should suggest why we have struggled to get lower bounds stronger than those for parity. More concretely, these techniques do not establish bounds stronger than $2^{\Omega(s(f)^{\frac{1}{d-1}})}$ where $s(f)$ is the sensitivity of $f : \{0,1\}^n \to \{0,1\}$. But $s(f) \leq n$, so this leaves us stuck the current state of the art ($2^{\Omega(\sqrt{n})}$ for depth-3).

To push beyond our current small-depth circuit lower bounds, there is a need for new techniques. In particular, we need to make use of complexity measures beyond sensitivity, where parity is no longer the hardest function. Moreover, to solve Open Problem 1, such a complexity measure needs to be one where majority is demonstrably harder than parity. The notion of composition complexity and the techniques of this paper meet both of these demands (as shown by Theorem 2). We are hopeful that these techniques (described in Section 1.3) can be extended to prove stronger lower bounds in more general settings.

Composition complexity and depth-3 circuits

If $\mathsf{CC}_k(f) \leq m$, then we can write $f = h(g_1, \ldots, g_m)$ where each of the functions g_j only needs to query k variables. But then we can write h as a DNF (or CNF) of size 2^m, and we can write the inner functions g_j as CNFs (or DNFs) of size 2^k. In this form we have a depth-3 circuit for f of size

$$2^m + m \cdot 2^k \leq 2^{O(m+k)} \tag{1}$$

with bottom fan-in k. The best-known depth-3 circuits for computing MAJ_n are obtained in precisely this manner, by using the bound $\mathsf{CC}_k(\mathrm{MAJ}_n) \leq O(\frac{n}{k}\log k)$ and setting $k := \sqrt{n\log n}$.

It follows that in order to prove that MAJ_n requires depth-3 circuits of size $2^{\Omega(\sqrt{n\log n})}$, one must first prove that $\max(\mathsf{CC}_k(f), k) \geq \Omega(\sqrt{n\log n})$. This is implied directly by Theorem 2, which we view as a key first step towards proving stronger depth-3 lower bounds. In particular,

---

[2] Briefly, in the notation of [31], it requires that $kn/c < n$, and so $k < c$, but c then takes on all the values of $n, n/2, n/3, \ldots, 1$ and $k \approx \sqrt{n}$. The third author thanks Srikanth Srinivasan [29] who informed him about this gap in the proof.

it shows that if one wanted to construct a depth-3 circuit for MAJ_n of size $2^{o(\sqrt{n \log n})}$, it would have to look very different from the current divide-and-conquer strategy. In Section 6.1 we outline some ways in which our results could be extended beyond the model we consider to depth-3.

1.3 Our techniques

In this section, we briefly describe the techniques we use to prove Theorem 2, and highlight some aspects that we feel are particularly interesting.

Information theory

While there have been some attempts to incorporate information theory into the toolbox of boolean function lower bounds [22, 8], these techniques remain uncommon. Our proof crucially uses *mutual information* to measure information flow from the input variables to the inner functions.

Our key insight can be summed up as the counterintuitive maxim "the less it is queried the more it is revealed". More concretely, suppose we can compute the Hamming weight function as $h(g_1, \ldots, g_m)$, then we show that if few of the inner functions $g_1, \ldots, g_m$ query a particular input variable x_i, then the output of the inner functions must reveal a lot of information about x_i's value – that is, one can guess x_i better than random chance based on $g_1(x), \ldots, g_m(x)$.

▶ **Lemma 4** (key insight, informally). *Suppose that x_i is queried by at most q of the inner functions $g_1, \ldots, g_m$. Then $\mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \geq 2^{-O(q)}$.*

One of the novelties of our proof is that the information flow can be distilled so cleanly as the above lemma, and that tight lower bounds follow quite easily from it (see Section 3). It is natural to wonder whether this approach can be generalized to stronger models of computation.

From multi-output functions to binary-output functions

Another intriguing aspect of our proof is that in proving the lower bound for majority, it turns out to be easiest to first prove a lower bound for the Hamming weight function $\text{HW}_n : \{0,1\}^n \to \{0, \ldots, n\}$, which maps $x \mapsto |x|$ (i.e., the number input bits that are 1). Notably, this is not a binary-output function, but rather a "multi-output" function (its output takes $\lceil \log(n+1) \rceil$ bits), and the proof of our information-theoretic Lemma 4 fundamentally uses this larger output space.

The way in which we extend the lower bound from HW_n to MAJ_n is also worth noting. In Section 4, we give a general framework for, in a sense, forcing binary-output functions to become multi-output. We use "control variables" to manipulate the construction $h(g_1, \ldots, g_m)$ into telling us more about the input, and "buffer variables" to avoid incurring a blowup in how many inner functions are necessary. This allows us to show that an efficient construction for MAJ_n would imply a similarly efficient construction computing the Hamming weight on a large fraction of inputs, and we can then conclude with a slightly more general version of Lemma 4.

Perhaps there is more to be found in this direction. Could the frontier of circuit lower bounds be pushed further by first proving lower bounds for multi-output functions, and then bootstrapping these to get lower bounds against the usual single-output functions? Indeed,

our proof technique suggests that proving more lower bounds for multi-output functions could be a valuable endeavor, even when those lower bounds do not seem to immediately lead to lower bounds for binary-output functions we traditionally study.

Natural proofs

The proof of our key lemma in Section 5 is tailored specifically to the Hamming weight function.[3] While this could be seen as a limitation, it can also be seen as a strength. Indeed, Razborov and Rudich [27] showed that lower bound arguments cannot apply to too broad a range of functions, assuming that pseudorandom functions exist. Given that our lower bound argument only works for the Hamming weight and closely related functions like the majority function, it does not seem to fall within the natural proofs framework.

1.4 Related work

Hrubeš-Rao and Nechiporuk's method

Hrubeš and Rao [15] gave a function f for which $\mathsf{CC}_k(f) \geq n^{\Omega(1-k/n)}$.[4] Their proof draws inspiration from Nechiporuk's method [21], which gives lower bounds for functions f for which one can split the variables into disjoint sets $S_1, \ldots, S_\ell$ such that f has many distinct subfunctions on each S_r $(r \in [\ell])$.

Their lower bound is quantitatively stronger than ours, but just like other bounds obtained from Nechiporuk's method, it only applies to a limited set of functions specially created for that purpose, and says nothing about many basic functions like majority. Nechiporuk's method is unable to prove lower bounds for majority because its subfunctions are all threshold functions, of which there are only a handful.[5] Moreover, the functions that [15] uses have no bearing on stronger lower bounds for depth-3 circuits, since their functions actually have depth-2 circuits of size $n^{O(\log n)}$.

It is interesting to contrast our techniques with Nechiporuk's method. At a very high level, Nechiporuk's method considers what one can infer about the function after fixing several variables (particularly, how many subfunctions remain), whereas our method considers what one can infer about the variables given the output of several of the inner functions. This difference in perspective is key in our ability to get tight lower bounds in a case where Nechiporuk's method only gives trivial bounds.

Lower bounds for canonical boolean circuits

Goldreich and Wigderson [11] recently introduced a new restricted of model of "canonical" boolean circuits, with the hope of proving $2^{\omega(\sqrt{n})}$ lower bounds for this model. Their model is inspired by the structure of optimal depth-3 circuits for $\textsc{Parity}_n$, which dissects the computation into disjoint parities of smaller arity (over $\sqrt{n}$ variables). [11] proposes a generalization of this construction, where one aims to represent a multi-linear function as a depth-2 circuit where the gates are *arbitrary multi-linear functions of small arity*. They

---

3 At a high level, it uses the facts that the possible Hamming weight values $0, \ldots, n$ are a completely ordered set and that flipping a bit from 0 to 1 increases the Hamming weight by one, in order to find one weight w^* for which the corresponding inputs are biased on a given coordinate.

4 Their paper denotes k-composition complexity as C_k^2 instead of CC_k.

5 Indeed, there is an exactly analogous situation with branching programs. While Nechiporuk's method can establish strong branching program lower bounds (see [26, Theorem 2], attributed to Beame and Cook), [2, 4] had to introduce new techniques to prove lower bounds for majority.

propose proving strong lower bounds in this model as a "sanity check" for proving better general depth-3 circuit bounds (Open Problem 1) – to do the latter, one must necessarily do the former as well.

Our approach can be viewed in a very similar light. In fact, our model is strictly stronger: we consider gates that compute arbitrary *boolean functions* of small arity, not just functions that are multi-linear over GF(2). In the same way, our model serves as a sanity check for Open Problem 1 – any proof that depth-3 circuits for MAJ_n require size $2^{\Omega(\sqrt{n \log n})}$ must prove Theorem 2 as well.

Strong lower bounds have indeed been proven in the model introduced by [11]. Goldreich and Tal [9] proved a lower bound of $2^{\tilde{\Omega}(n^{2/3})}$. [10] also proved lower bounds of $2^{\tilde{\Omega}(n^{3/8})}$ for canonical depth-4 circuits, an improvement over the $2^{\Omega(n^{1/3})}$ bound that is known for the general depth-4 circuits computing PARITY_n. In our model where the gates can compute arbitrary boolean functions of small arity, such strong bounds are not possible for the MAJ_n function, and we pin down exactly the right bounds in this case.

Majority as a majority of majorities

There has been interest [18, 6, 14, 3, 25] in the optimal ways of computing majority as a composition of functions $h(g_1, \ldots, g_m)$ in the restricted model, where the h and $g_1, \ldots, g_m$ are majority functions of smaller fan-in ($\text{MAJ}_{\leq k}$ for some k). [18] showed that $k \geq n^{0.7}$ was necessary, [6] improved this to $k \geq n^{0.8}$, and [14] further improved this to $k \geq n/2 - o(n)$. In terms of upper bounds, [3] gave $k \leq n+2$ for odd $n \geq 7$, and [25] improved this to $k \leq \frac{2}{3}n+4$ (for all n).

Similar to the state of canonical boolean circuits discussed above, this setting is more restrictive, so it makes sense that their lower bounds are stronger than Theorem 2. It would be interesting to see if our techniques could apply to these more restricted models as well.

2 Preliminaries

2.1 Locality

In this subsection, we define some notation that we will use repeatedly, and we give a formal definition of k-locality.

▶ **Definition 5** ($x^{(i \mapsto b)}$, $x^{\oplus i}$)**.** *For any $x \in \{0,1\}^n$, $i \in [n]$ and $b \in \{0,1\}$, let*
- $x^{(i \mapsto b)} := (x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$;
- $x^{\oplus i} := (x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_n)$.

▶ **Definition 6** (k-local)**.** *A function $g : \{0,1\}^n \to \{0,1\}$ is k-local if there is a set of variables $I \subseteq [n]$ with $|I| = k$ such that g depends only on the variables in I, i.e. for each $x \in \{0,1\}^n$ and each $i \in [n] \setminus I$, we have $g(x) = g(x^{\oplus i})$.*

▶ **Definition 7** (composition complexity, formal version of Definition 1)**.** *The k-composition complexity of a function $f : \{0,1\}^n \to D$, denoted $\mathsf{CC}_k(f)$, is the minimum integer m such that there exist functions $g_1, \ldots, g_m : \{0,1\}^n \to \{0,1\}$ and $h : \{0,1\}^m \to D$ with the following properties:*
  (i) *for all $x \in \{0,1\}^n$, $f(x) = h(g_1(x), \ldots, g_m(x))$;*
  (ii) *for each $j \in [m]$, g_j is k-local.*

▶ **Remark 8.** Note that the inner functions g_j are restricted to having binary outputs. This is necessary for making the definition nontrivial: if their output domains were arbitrary, then the composition complexity would always be $O(n/k)$, since we could simply let each inner function output the values of all of the variables they query.

2.2 Number of queries per variable

In this subsection, we show that without loss of generality, we can assume that all variables are queried roughly the same number of times. More precisely, say that $f = h(g_1, \ldots, g_m)$ where each g_j is k-local. Then the total number of queries is at most mk, so the average variable is queried at most $\frac{mk}{n}$ times. We will show that for the purpose of proving lower bounds on composition complexity, we can assume that *every* variable is queried at most $\frac{mk}{n}$ times.

▶ **Definition 9** (self-containing). *A family of functions* $\{f_n\}_{n \in \mathbb{N}}$ *is* self-containing *if for any n and any $I \subseteq [n]$, there is a subfunction of f_n on I that computes $f_{|I|}$.*

▶ **Fact 10.** *Both majority* $\{\mathrm{MAJ}_n\}_{n \in \mathbb{N}}$ *and Hamming weight* $\{\mathrm{HW}_n\}_{n \in \mathbb{N}}$ *are self-containing.*

▶ **Lemma 11.** *Let* $\{f_n\}_{n \in \mathbb{N}}$ *be a self-containing family of functions. Suppose that* $\mathsf{CC}_k(f_{2n}) \leq m$. *Then we can write* $f_n = h(g_1, \ldots, g_m)$ *where each g_j is k-local and each variable is queried at most $\frac{mk}{n}$ times.*

Proof. Suppose $f_{2n} = h(g_1, \ldots, g_m)$, where each g_j is k-local. Let $q := \frac{mk}{2n}$. Then the average variable is queried $\leq q$ times, so by Markov's inequality at most half of the variables are queried more than $2q$ times. Let I be any set of n variables, each of which is queried at most $2q = \frac{mk}{n}$ times. Since $\{f_n\}_{n \in \mathbb{N}}$ is self-containing, there is a subfunction of f_{2n} on I that computes f_n. The lemma follows by restricting each g_j to I. ◀

▶ **Corollary 12.** *In order to prove that* $\mathsf{CC}_k(f_n) \geq \Omega(\frac{n}{k} \log k)$, *it is enough to prove that if* $f_n = h(g_1, \ldots, g_m)$ *and each variable is queried at most $\frac{mk}{n}$ times by the inner functions g_j, then $m \geq \Omega(\frac{n}{k} \log k)$.*

▶ **Remark 13.** The quantity $\frac{mk}{n}$ corresponds exactly to our definition of composition overhead (recall the discussion following Definition 1). This makes sense, since the composition overhead is the ratio of how many inner functions we need compared to the ideal situation where each variable is queried exactly once. The more inner functions, the more queries per variable (assuming they all query roughly k variables).

2.3 Information theory

In this subsection, we introduce some information-theoretic notions and properties that are used in our proofs. To learn more about information theory from a theoretical computer science perspective, we recommend checking out the Simons Institute workshop titled "Information Theory Boot Camp".[6]

The most important notion in information theory is the *entropy* of a random variable $\boldsymbol{X}$, denoted $\mathbf{H}[\boldsymbol{X}]$, which represents "how much randomness" the variable contains, or how many bits I need to communicate to you on average for you to learn $\boldsymbol{X}$.

▶ **Definition 14** (entropy). *Given a random variable* $\boldsymbol{X}$ *with support D, the* entropy of $\boldsymbol{X}$ *is the quantity*

$$\mathbf{H}[\boldsymbol{X}] := \sum_{x \in D} \mathbf{Pr}[\boldsymbol{X} = x] \log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = x]}\right) = \underset{\boldsymbol{X}' \sim \boldsymbol{X}}{\mathbf{E}}\left[\log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = \boldsymbol{X}']}\right)\right]$$

where $\boldsymbol{X}'$ is an independent copy of $\boldsymbol{X}$.

---

[6] https://simons.berkeley.edu/workshops/inftheory2015-boot-camp

A related notion is the *conditional entropy* $\mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}]$ of two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, which represents the "how much randomness remains" in $\boldsymbol{X}$ once you know $\boldsymbol{Y}$, or how many bits I need to communicate to you on average for you to learn $\boldsymbol{X}$, assuming that you already know $\boldsymbol{Y}$.

▶ **Definition 15** (conditional entropy). *Given two random variables* $(\boldsymbol{X}, \boldsymbol{Y})$ *over domain D, the* entropy of $\boldsymbol{X}$ conditioned on $\boldsymbol{Y}$ *is the quantity*

$$\mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}] := \sum_{(x,y) \in D} \mathbf{Pr}[\boldsymbol{X} = x \wedge \boldsymbol{Y} = y] \log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = x \mid \boldsymbol{Y} = y]}\right)$$

$$= \mathop{\mathbf{E}}_{(\boldsymbol{X}', \boldsymbol{Y}') \sim (\boldsymbol{X}, \boldsymbol{Y})}\left[\log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = \boldsymbol{X}' \mid \boldsymbol{Y} = \boldsymbol{Y}']}\right)\right]$$

where $(\boldsymbol{X}', \boldsymbol{Y}')$ *is an independent copy of* $(\boldsymbol{X}, \boldsymbol{Y})$.

The entropy and conditional entropy have the following properties, which we use in our proofs.

▶ **Fact 16** (bounds). *If $\boldsymbol{X}$ is a random variable over a finite domain D and $\boldsymbol{Y}$ is a random variable, then*

$$0 \le \mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}] \le \mathbf{H}[\boldsymbol{X}] \le \log|D|,$$

where the last inequality is tight iff $\boldsymbol{X}$ is uniform on D.

▶ **Fact 17** (subadditivity). *Let $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ be two random variables. Then $\mathbf{H}[\boldsymbol{X}_1, \boldsymbol{X}_2] \le \mathbf{H}[\boldsymbol{X}_1] + \mathbf{H}[\boldsymbol{X}_2]$, with equality iff $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are independent. Similarly, $\mathbf{H}[\boldsymbol{X}_1, \boldsymbol{X}_2 \mid \boldsymbol{Y}] \le \mathbf{H}[\boldsymbol{X}_1 \mid \boldsymbol{Y}] + \mathbf{H}[\boldsymbol{X}_2 \mid \boldsymbol{Y}]$.*

▶ **Fact 18** (dependence). *Let $\boldsymbol{X}, \boldsymbol{Y}$ be random variables such that $\boldsymbol{X}_2$ is completely determined by $\boldsymbol{X}_1$ (i.e. $\boldsymbol{X}_2 = f(\boldsymbol{X}_1)$ where f is a function). Then we have the following:*
- $\mathbf{H}[\boldsymbol{X}_2 \mid \boldsymbol{X}_1] = 0$;
- $\mathbf{H}[\boldsymbol{Y} \mid \boldsymbol{X}_1] \le \mathbf{H}[\boldsymbol{Y} \mid \boldsymbol{X}_2]$.

Finally, a crucial notion in our proof is the *mutual information* $\mathbf{I}[\boldsymbol{X} : \boldsymbol{Y}]$ of two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, which represents "how much information $\boldsymbol{X}$ reveals about $\boldsymbol{Y}$", or symmetrically, "how much information $\boldsymbol{Y}$ reveals about $\boldsymbol{X}$".

▶ **Definition 19** (mutual information). *Given two random variables $\boldsymbol{X}, \boldsymbol{Y}$, the* mutual information *of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the quantity $\mathbf{I}[\boldsymbol{X} : \boldsymbol{Y}] := \mathbf{H}[\boldsymbol{X}] - \mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}]$.*

▶ **Fact 20** (symmetry of mutual information). $\mathbf{I}[\boldsymbol{X} : \boldsymbol{Y}] = \mathbf{I}[\boldsymbol{Y} : \boldsymbol{X}]$.

3 The less it is queried, the more it is revealed

Hamming weight: a multi-output function

Even though the main function of interest in this paper is the majority function, we will first prove a lower bound for the related *Hamming weight* function: a "multi-output" function (as opposed to binary-output) that reveals the entire Hamming weight of the input string.

▶ **Definition 21.** *Let* $\mathrm{HW}_n : \{0,1\}^n \to \{0, 1, \ldots, n\} : x \mapsto |x| = x_1 + \cdots + x_n$ *be the Hamming weight function.*

The most natural way to express HW_n as $h(g_1, \ldots, g_m)$ is to split the n variables into groups of k variables, and for each group to create $\lceil \log(k+1) \rceil$ inner functions g_j, each computing one bit of the sum of the k variables in this group. The function h can then compute the Hamming weight of the whole input string by first recovering the sum for each group, then adding them up. Clearly, each of the inner functions is k-local, and there are $O(\frac{n}{k} \log k)$ of them, so $\mathsf{CC}_k(\mathrm{HW}_n) \leq O(\frac{n}{k} \log k)$. We will show that this is optimal: $\mathsf{CC}_k(\mathrm{HW}_n) = \Theta(\frac{n}{k} \log k)$.

Simple counting does not give much

It is easy to see $\mathsf{CC}_k(\mathrm{HW}_n) > n/k$: suppose that there were a way to represent HW_n as $h(g_1, \ldots, g_m)$ with $m = n/k$. Then each of the inner functions g_j would query k variables and each variable would be queried only once. Take g_1, and consider the k variables it queries. Say that we fix all other variables to 0. Then there are still $k+1$ possible values for the Hamming weight. But this fixes the outputs of $g_2, \ldots, g_m$, so there can only be two possible values for $h(g_1, \ldots, g_m)$ (one where g_1 outputs 0 and the other where g_1 outputs 1), so we have a contradiction.

In general, it is easy to see that each set of r variables must collectively be queried by at least $\log(r+1)$ different inner functions. But this observation is not enough to show that the average variable will need to be queried a super-constant number of times.

A very counterintuitive lemma

Basic counting arguments like the above do not seem to say anything that keeps a variable from being queried by only a constant number of inner functions. But it turns out there *is* something we can say about variables that are queried by few inner functions: the outputs of the inner functions must reveal a lot about their value, in terms of mutual information. Put another way, this means that if you are given the outputs of the inner functions, you can often guess the value of such variables better than random chance.

▶ **Lemma 22** (key lemma). *Suppose that* $\mathrm{HW}_n = h(g_1, \ldots, g_m)$ *and that variable i is queried at most q of the inner functions $g_1, \ldots, g_m$. Let $\boldsymbol{X} \sim \{0,1\}^n$ be a uniformly random input. Then* $\mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \geq 2^{-O(q)}$.

Consider how counterintuitive this statement is: it says that the *less* the ith variable is queried by the inner functions $g_1, \ldots, g_m$, the *more* it will be revealed by their collective outputs.

A lower bound for Hamming weight, assuming Lemma 22

We will not prove Lemma 22 until Section 5, but let us see how it implies the lower bound we want. First, we will show the intuitive fact that "if the inner functions reveal a lot about many of the variables, then there must be many inner functions".

▶ **Corollary 23.** *Suppose that* $\mathrm{HW}_n = h(g_1, \ldots, g_m)$ *and each variable is queried by at most q of the inner functions $g_1, \ldots, g_m$. Then* $m \geq n \cdot 2^{-O(q)}$.

Proof. We bound the quantity $\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})]$ in two ways. First, we give an upper bound on m based on the fact that there are only m inner functions, each of which outputs a single bit. Then we lower bound this same quantity using Lemma 22. Combining these bounds yields the corollary.

On the one hand,

$$
\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] &= \mathbf{H}[g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] - \mathbf{H}[g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X}) \mid \boldsymbol{X}] \\
&= \mathbf{H}[g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\qquad\qquad\qquad\text{(because $\boldsymbol{X}$ determines $g_j(\boldsymbol{X})$ completely)} \\
&\leq \mathbf{H}[g_1(\boldsymbol{X})] + \ldots + \mathbf{H}[g_m(\boldsymbol{X})] \quad \text{(because $\mathbf{H}[\cdot]$ is subadditive)} \\
&\leq m. \qquad\qquad\qquad \text{(because each g_j has a binary output)}
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] &= \mathbf{H}[\boldsymbol{X}] - \mathbf{H}[\boldsymbol{X} \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&= \left(\sum_{i \in [n]} \mathbf{H}[\boldsymbol{X}_i] \right) - \mathbf{H}[\boldsymbol{X} \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\qquad\qquad\qquad \text{(because the $\boldsymbol{X}_i$ are independent)} \\
&\geq \sum_{i \in [n]} (\mathbf{H}[\boldsymbol{X}_i] - \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})]) \\
&\qquad\qquad\qquad \text{(because $\mathbf{H}[\cdot \mid \boldsymbol{Y}]$ is subadditive for any $\boldsymbol{Y}$)} \\
&= \sum_{i \in [n]} \mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\geq \sum_{i \in [n]} 2^{-O(q)} \qquad\qquad\qquad \text{(by Lemma 22)} \\
&= n \cdot 2^{-O(q)}.
\end{aligned}
$$

The corollary follows from combining these two bounds. ◄

We can now use Corollary 23 to deduce the desired lower bound.

▶ **Theorem 24** (composition complexity of HW_n). $\mathsf{CC}_k(\mathrm{HW}_n) \geq \Omega(\frac{n}{k} \log k)$.

Proof. Suppose that $\mathrm{HW}_n = f(g_1, \ldots, g_m)$, where each of the inner functions $g_1, \ldots, g_m$ is k-local. By Corollary 12, we only have to show that $m \geq \Omega(\frac{n}{k} \log k)$ under the assumption that each variable is queried at most $q := \frac{mk}{n}$ times. Then by Corollary 23,

$$
\frac{nq}{k} = m \geq n \cdot 2^{-O(q)}.
$$

Rearranging, we get

$$
q \cdot 2^{O(q)} \geq k \Rightarrow q \geq \Omega(\log k),
$$

which means $m = \frac{n}{k} q \geq \Omega(\frac{n}{k} \log k)$. ◄

4 From Hamming weight to majority

In the previous section, we gave a lower bound on the composition complexity for the Hamming weight function HW_n. This is a multi-output function (it has more than two possible outputs), and the proof arguably relied a lot on this fact. In this section, we show how to extend the lower bound to MAJ_n, a binary-output function, by manipulating it to become multi-output.

▶ **Definition 25** (majority function). *Let*

$$\text{MAJ}_n : \{0,1\}^n \to \{0,1\} : x \mapsto \begin{cases} 1 & \text{if } |x| \geq n/2 \\ 0 & \text{otherwise} \end{cases}$$

be the majority function.

4.1 Reduction toolkit

Before proving the lower bound, let us introduce the tools we will use to reduce majority to Hamming weight.

Control variables

$\text{MAJ}_n(x)$ tells us whether x has Hamming weight $\geq n/2$ or $< n/2$, but nothing more. For example, what if we wanted it to also tell us something special when the Hamming weight is *exactly* $n/2$? Suppose n is even, then we could figure this out by looking at the values of $\text{MAJ}_{n+1}(x_1, \ldots, x_n, 0)$ and $\text{MAJ}_{n+1}(x_1, \ldots, x_n, 1)$: indeed,

- if $|x| < n/2$ then both return 0;
- if $|x| > n/2$ then both return 1;
- if $|x| = n/2$ then $\text{MAJ}_{n+1}(x_1, \ldots, x_n, 0) = 0$ but $\text{MAJ}_{n+1}(x_1, \ldots, x_n, 1) = 1$.

Note how the last variable of MAJ_{n+1} does not stay free, but rather is assigned a fixed value that modifies the behavior of MAJ_{n+1} as a function of $x_1, \ldots, x_n$. For this reason, we will call it a *control variable*, and we will call the n other variables *free variables*.

Now, assume that MAJ_{n+1} can be computed as $h(g_1, \ldots, g_m)$. Then we can compute the function

$$f : \{0,1\}^n \to \{0,1,2\} : x \mapsto \begin{cases} 0 & \text{if } |x| < n/2 \\ 1 & \text{if } |x| = n/2 \\ 2 & \text{if } |x| > n/2 \end{cases}$$

as $h'(g_1^0, \ldots, g_l^0, g_1^1, \ldots, g_l^1)$ where

$$g_j^0(x) := g_j(x_1, \ldots, x_n, 0) \qquad g_j^1(x) := g_j(x_1, \ldots, x_n, 1).$$

This suggests a potential approach for proving a lower bound on the composition complexity MAJ_n: add enough control variables to it so that the values we get are enough to determine the Hamming weight of x, then use the lower bound for HW_n as a black box. Indeed, if we know all of the values

$$\begin{cases} \text{MAJ}_{2n-1}(x_1, \ldots, x_n, 0, 0, \ldots, 0) \\ \text{MAJ}_{2n-1}(x_1, \ldots, x_n, 1, 0, \ldots, 0) \\ \qquad\qquad \vdots \\ \text{MAJ}_{2n-1}(x_1, \ldots, x_n, 1, 1, \ldots, 1) \end{cases}$$

then we can recover $|x|$. However, this would cause a huge blowup in the number of inner functions needed to compute it: assuming MAJ_{2n-1} can be computed as $h(g_1, \ldots, g_m)$, this would only guarantee that HW_n can be computed as a function of the nm components

$$g_j(x_1, \ldots, x_n, 0, \ldots, 0), \ldots, g_j(x_1, \ldots, x_n, 1, \ldots, 1)$$

for $j \in [m]$, thus the $\Omega(\frac{n}{k} \log k)$ lower bound we have for HW_n would fail to give any nontrivial lower bound on m. So we need to do this in a smarter way.

Buffer variables

We will avoid this blowup by using some *buffer variables* to "isolate" the control variables from the free variables. Assume that $\mathrm{MAJ}_n = h(g_1, \ldots, g_m)$. Suppose we split the variables into three sets $I_{\text{free}}, I_{\text{control}}, I_{\text{buffer}} \subseteq [n]$ such that none of the inner functions g_j queries variables from both I_{free} and I_{control} (informally, I_{buffer} acts as a buffer between I_{free} and I_{control}; see Figure 1).
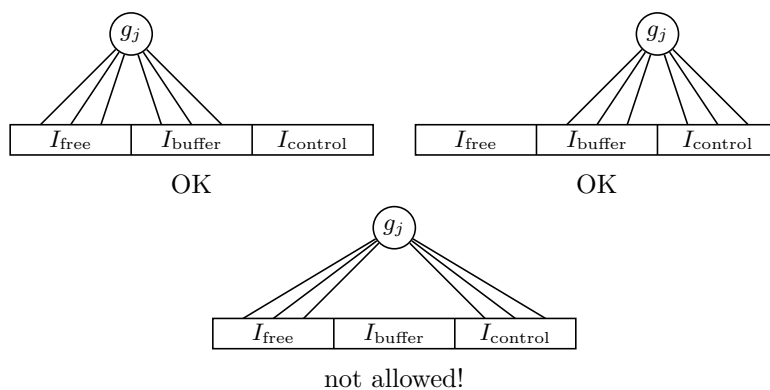
Figure 1 The inner functions g_j are allowed to query variables in both I_{free} and I_{buffer}, or variables in both I_{buffer} and I_{control}, but not variables in both I_{free} and I_{control}.

Then if we fix the value of the variables in I_{buffer}, we can avoid a blow up in the number of times each variable is queried. Let us informally see why. There are two types of inner functions g_j:

- if g_j queries a free variable, then it does not query any control variables, so no blowup will happen;
- if g_j does *not* query any free variables, then it only depends on the control variables and the buffer variables, both of which are known, so in that case we can recompute the output of g_j ourselves.

A bit more formally, let $x^{(I_{\text{control}} \mapsto 0)}$ denote x with all variables in I_{control} set to 0. Then we can recover $\mathrm{MAJ}_n(x)$ if we know only $g_1(x^{(I_{\text{control}} \mapsto 0)}), \ldots, g_m(x^{(I_{\text{control}} \mapsto 0)})$ as well as the value of the variables in I_{control} and I_{buffer}. Since $g_j(x^{(I_{\text{control}} \mapsto 0)})$ does not depend on the value of the variables in I_{control}, no blowup happens when we start to vary their values.

Partial functions

Even with the help of buffer variables, though, it turns out we will not be able to have enough control variables to compute HW_n from MAJ_n.[7] We will only be able to compute the Hamming weight correctly on a fraction of the possible inputs. Therefore we need to adapt our techniques from Section 3 to a "partial functions" setting, where $f(x) = |x|$ is only guaranteed when x is in a subset $D \subseteq \{0,1\}^n$ of the possible inputs.

First, the following lemma is a generalization of our key lemma Lemma 22, which gave a lower bound on the mutual information between $\boldsymbol{X}_i$ and the inner function outputs $g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})$ when variable i is queried few times. What differs from Lemma 22 is that

---

[7] Or more precisely, to compute $\mathrm{HW}_{n'}$ from MAJ_n for some $n' = \Omega(n)$.

the result gives a nontrivial bound only when the probability $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D]$ is small. This probability measures how likely you are to leave D if you start out in D then flip the ith bit, so intuitively, the bigger D is, the smaller it will be.

▶ **Lemma 26** (key lemma, "partial functions" version). *Let $D \subseteq \{0,1\}^n$ be a non-empty subset of the hypercube (the "domain" on which f computes the Hamming weight), and let $f : \{0,1\}^n \to \{0,\ldots,n\}$ be a function such that $\forall x \in D, f(x) = |x|$. Suppose that $f = h(g_1,\ldots,g_m)$ and that variable i is queried by at most q of the inner functions $g_1,\ldots,g_m$. Let $\boldsymbol{X} \sim D$ be a random input drawn uniformly over D. Then $\mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}$.*

We postpone the proof of this lemma until Section 5. For now, let us see why it implies Lemma 22.

Proof of Lemma 22 assuming Lemma 26. Set $f \coloneqq \mathrm{HW}_n$. Then $f(x) = |x|$ for all $x \in \{0,1\}^n$, so we can set $D \coloneqq \{0,1\}^n$. This means that $\boldsymbol{X}$ is a uniformly random input, and $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] = 0$. Therefore, after rearranging, we obtain

$$
\begin{aligned}
2^{-O(q)} &\leq 1 - \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \\
&= \mathbf{H}[\boldsymbol{X}_i] - \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] && \text{(because } \boldsymbol{X} \text{ is uniformly random)} \\
&= \mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})]. && \text{(by definition of mutual information)}
\end{aligned}
$$

◀

We can also prove an analog of Corollary 23, which gave a lower bound on the number of inner functions m assuming that many variables are queried few times by the inner functions.

▶ **Corollary 27.** *There is an integer constant $C > 0$ such that the following holds. Let $D \subseteq \{0,1\}^n$ be a non-empty subset of the hypercube, and let $f : \{0,1\}^n \to \{0,\ldots,n\}$ be a function such that $\forall x \in D, f(x) = |x|$. Suppose that $f = h(g_1,\ldots,g_m)$, that each variable is queried by at most $q > 0$ of the inner functions $g_1,\ldots,g_m$, and that $\mathbf{Pr}_{\boldsymbol{X} \sim D}[\boldsymbol{X}^{\oplus i} \notin D] \leq 2^{-Cq}$. Then $m + (n - \log|D|) \geq n \cdot 2^{-O(q)}$.*

Proof. Let C' be a possible integer value for the constant in the $O(\cdot)$ of Lemma 26. We will prove the corollary for $C \coloneqq C' + 1$. The proof works in the same way as the proof of Corollary 23: we will upper and lower bound the quantity $\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})]$. On the one hand,

$$
\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] &= \mathbf{H}[g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] - \mathbf{H}[g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X}) \mid \boldsymbol{X}] \\
&= \mathbf{H}[g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \\
& \qquad\qquad\qquad \text{(because } \boldsymbol{X} \text{ determines } g_j(\boldsymbol{X}) \text{ completely)} \\
&\leq \mathbf{H}[g_1(\boldsymbol{X})] + \ldots + \mathbf{H}[g_m(\boldsymbol{X})] \quad \text{(because } \mathbf{H}[\cdot] \text{ is subadditive)} \\
&\leq m. \qquad\qquad\qquad\qquad\quad \text{(because each } g_j \text{ has a binary output)}
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] &= \mathbf{H}[\boldsymbol{X}] - \mathbf{H}[\boldsymbol{X} \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\geq \mathbf{H}[\boldsymbol{X}] - \sum_{i \in [n]} \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\qquad\qquad \text{(because } \mathbf{H}[\cdot \mid \boldsymbol{Y}] \text{ is subadditive for any } \boldsymbol{Y}) \\
&\geq \mathbf{H}[\boldsymbol{X}] - \sum_{i \in [n]} \left(1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-C'q} \right) \quad \text{(by Lemma 26)} \\
&\geq \mathbf{H}[\boldsymbol{X}] - n + \sum_{i \in [n]} \left(2^{-C'q} - 2^{-(C'+1)q} \right) \\
&\qquad\qquad \text{(by assumption that } \mathbf{Pr}_{\boldsymbol{X} \sim D}[\boldsymbol{X}^{\oplus i} \notin D] \leq 2^{-Cq}) \\
&= \log |D| - n + \sum_{i \in [n]} 2^{-O(q)}. \\
&\qquad\qquad \text{(because } \boldsymbol{X} \text{ is uniform on } D, \text{ and because } q > 0)
\end{aligned}
$$

The corollary follows from combining these two bounds. ◀

4.2 A lower bound for majority

With all the tools in hand, let us prove a lower bound on the composition for MAJ_n by reducing it to a partial version of Hamming weight.

▶ **Theorem 28** (composition complexity of MAJ_n). $\mathsf{CC}_k(\mathrm{MAJ}_n) \geq \Omega(\frac{n}{k} \cdot \min(\log k, \log \frac{n}{k}))$.

Proof. Suppose that $\mathrm{MAJ}_n = h(g_1, \ldots, g_m)$ where each of the inner functions g_j is k-local. By Corollary 12, it is enough to show that $m \geq \Omega(\frac{n}{k} \cdot \min(\log k, \log \frac{n}{k}))$ under the assumption that each variable is queried at most $q := \frac{mk}{n}$ times. To do this, we will show that $q \geq \Omega(\min(\log k, \log \frac{n}{k}))$.

The plan of the proof is as follows. We will start by defining the sets of variables I_{free}, I_{buffer} and I_{control}. We will make sure that $|I_{\mathrm{free}}| = \Omega(n)$, and we will play with the variables in I_{control} to show that the inner functions $g_1, \ldots, g_m$ must compute the Hamming weight over a large subset D of $\{0,1\}^{I_{\mathrm{free}}}$. From there, we will apply Corollary 27, and only asymptotic calculations will remain, similar to the proof of Theorem 24.

Defining I_{free}, I_{buffer} and I_{control}. Let C be the constant in Corollary 27. Assume that $2^{2Cq+1}qk \leq n/2$ (otherwise, $q = \Omega(\log \frac{n}{k})$ and we are done). Let I_{control} be any 2^{2Cq+1}-element subset of $[n]$, let $J \subseteq [m]$ be the set of inner functions that query some variable in I_{control}, and let $I' \subseteq [n]$ be the set of all variables queried by at least one of the inner functions in J. That is, let I' be the set of variables that are queried by some inner function g_j that also queries a variable in I_{control}.

Since each variable in I_{control} is queried by at most q inner functions, and each inner function queries at most k variables, we have $|I'| \leq |I| \cdot q \cdot k = 2^{2Cq+1}qk \leq n/2$. In case $|I'| < 2|I_{\mathrm{control}}|+1$, extend I' to include some more variables of $[n]$ until it reaches size $2|I_{\mathrm{control}}|+1$. After this, we still have $|I'| \leq \max(n/2, 2|I_{\mathrm{control}}|+1) = \max(n/2, 2 \cdot 2^{2Cq+1}+1) \leq n/2$.

Let $I_{\mathrm{free}} := [n] \setminus I'$, and let $I_{\mathrm{buffer}} := I' \setminus I_{\mathrm{control}}$ (see Figure 2). Note that $I_{\mathrm{free}} \cup I_{\mathrm{buffer}} \cup I_{\mathrm{control}} = [n]$. Since I_{free} contains no element of I', it is clear that no inner function queries both variables in I_{free} and I_{control}. Also, defining $n_{\mathrm{free}} := |I_{\mathrm{free}}|$, we have

$$
n_{\mathrm{free}} = n - |I'| \geq n - n/2 = n/2. \tag{2}
$$
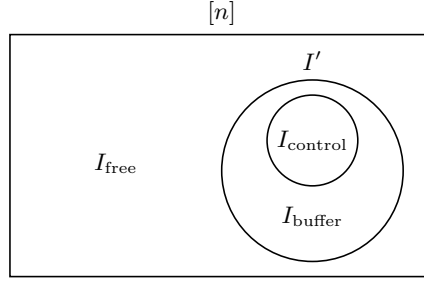
Figure 2 A schematic view of the sets of variables I_{free}, I_{buffer}, and I_{control}.

Computing a partial version of $\text{HW}_{n_{\text{free}}}$ from MAJ_n.   We will split the input into three parts $x = x_{\text{free}} \circ x_{\text{buffer}} \circ x_{\text{control}}$, where $x_{\text{free}} \in \{0,1\}^{I_{\text{free}}}$, $x_{\text{buffer}} \in \{0,1\}^{I_{\text{buffer}}}$ and $x_{\text{control}} \in \{0,1\}^{I_{\text{control}}}$. First, let us fix a value for x_{buffer}. We will fix x_{buffer} to be some vector with exactly $|x_{\text{buffer}}| = \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$ ones. To make sure this is possible, we need to show $0 \leq \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} \leq |I_{\text{buffer}}|$. Firstly, we have that $n - n_{\text{free}} = |I_{\text{buffer}}| + |I_{\text{control}}| \geq |I_{\text{control}}| = 2^{2Cq+1}$, so $\lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil \geq \lfloor \frac{n - n_{\text{free}}}{2} \rfloor \geq \lfloor \frac{2^{2Cq+1}}{2} \rfloor = 2^{2Cq}$, which gives the lower bound. Secondly, we have $|I_{\text{buffer}}| \geq |I'| - |I_{\text{control}}| \geq |I_{\text{control}}| + 1$ (since we made sure that $|I'| \geq 2|I_{\text{control}}| + 1$) and thus

$$|I_{\text{buffer}}| \geq \frac{|I_{\text{buffer}}| + |I_{\text{control}}| + 1}{2} = \frac{n - |I_{\text{free}}| + 1}{2} = \frac{n - n_{\text{free}} + 1}{2} \geq \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil,$$

which gives the upper bound.

Since each inner function $g_j(x)$ is either completely determined by x_{free} and x_{buffer} or completely determined by x_{buffer} and x_{control}, in order to compute $\text{MAJ}_n(x)$, it is enough to know $g_j(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}})$ for all $j \in [m]$ (where $0^{I_{\text{control}}} \in \{0,1\}^{I_{\text{control}}}$ is the all zeros string) as well as the value of x_{buffer} and x_{control}. So even if $g_j(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}})$ for $j \in [m]$ is all we know about x_{free}, we can compute the value of $\text{MAJ}_n(x_{\text{free}} \circ x_{\text{buffer}} \circ x_{\text{control}})$ for any x_{control} we desire. Thus, setting $|x_{\text{control}}|$ between 0 and $|I_{\text{control}}| = 2^{2Cq+1}$, we can figure out:

- whether $|x_{\text{free}}| + |x_{\text{buffer}}| + 0 \geq \lceil n/2 \rceil$;

- whether $|x_{\text{free}}| + |x_{\text{buffer}}| + 1 \geq \lceil n/2 \rceil$;

- ...

- whether $|x_{\text{free}}| + |x_{\text{buffer}}| + 2^{2Cq+1} \geq \lceil n/2 \rceil$.

Given that we set $|x_{\text{buffer}}| = \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$, this means we can distinguish between the following cases:

- $|x_{\text{free}}| \geq \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}$;

- $|x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} - 1$;

- ...

- $|x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$;

- $|x_{\text{free}}| < \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$.

Therefore, we can form a function $f(x_{\text{free}}) = h'(g_1(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}}), \dots, g_m(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}}))$ such that

$$f(x_{\text{free}}) = \begin{cases} \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} & \text{if } |x_{\text{free}}| \geq \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} \\ \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} - 1 & \text{if } |x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} - 1 \\ \dots \\ \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} & \text{if } |x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} \\ \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1 & \text{if } |x_{\text{free}}| < \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}. \end{cases}$$

Applying Corollary 27. This function f computes the Hamming weight correctly on the set

$$D = \{x_{\text{free}} \in \{0,1\}^{I_{\text{free}}} \mid \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1 \leq |x_{\text{free}}| \leq \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}\}.$$

We now prepare to apply Corollary 27 to f. Clearly, each input variable of f is queried by at most q of the inner functions $g_1(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}}), \dots, g_m(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}})$. Let us see check that D satisfies the condition of Corollary 27. Observe that $|\boldsymbol{X}^{\oplus i}| = |\boldsymbol{X}| \pm 1$, so when $\lceil n_{\text{free}}/2 \rceil - 2^{2Cq} \leq |\boldsymbol{X}| < \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}$, we have $\boldsymbol{X}^{\oplus i} \in D$. Therefore,

$$\Pr_{\boldsymbol{X} \sim D}[\boldsymbol{X}^{\oplus i} \notin D] \leq \Pr_{\boldsymbol{X} \sim D}[|\boldsymbol{X}| = \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1 \vee |\boldsymbol{X}| = \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}]$$

$$\leq \frac{2}{\#\{\lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1, \dots, \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}\}}$$

(because the further $|\boldsymbol{X}|$ is from $n_{\text{free}}/2$, the fewer possible values there are for $\boldsymbol{X}$)

$$= \frac{2}{2^{2Cq+1} + 2}$$

$$\leq 2^{-2Cq}.$$

Thus we can apply Corollary 27 and obtain

$$m + (n_{\text{free}} - \log|D|) \geq n_{\text{free}} \cdot 2^{-O(q)}. \tag{3}$$

Now, we also have

$$\log|D| \geq \log|\{x_{\text{free}} \in \{0,1\}^{I_{\text{free}}} \mid |x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil\}|$$

$$\geq \log(\Omega(2^{n_{\text{free}}}/\sqrt{n_{\text{free}}}))$$

$$= n_{\text{free}} - O(\log n_{\text{free}}),$$

so $n_{\text{free}} - \log|D| = O(\log n_{\text{free}}) = O(\log n)$. Plugging this into (3), we get

$$m + O(\log n) \geq n_{\text{free}} \cdot 2^{-O(q)}. \tag{4}$$

Asymptotics and conclusion. Since each inner function $g_1, \dots, g_m$ queries k variables and each variable is queried at most q times, we have $mk \leq nq$. Therefore,

$$\frac{nq}{k} + O(\log n) \geq m + O(\log n)$$

$$\geq n_{\text{free}} \cdot 2^{-O(q)} \tag{by (4)}$$

$$\geq \frac{n}{2} \cdot 2^{-O(q)}. \tag{by (2)}$$

Rearranging, we get $2^{O(q)}\left(\frac{q}{k} + \frac{O(\log n)}{n}\right) \geq 1/2$, so either

$$\frac{2^{O(q)}q}{k} \geq 1/4 \Rightarrow q = \Omega(\log k)$$

or

$$\frac{2^{O(q)}O(\log n)}{n} \geq 1/4 \Rightarrow q = \Omega(\log n) \geq \Omega(\log k). \qquad \blacktriangleleft$$

5   Proof of Lemma 26

In this section we present the proof of our key lemma Lemma 26, which is a generalization of Lemma 22 to functions that only compute the Hamming weight correctly on a subset of the inputs.

▶ **Lemma 26** (key lemma, "partial functions" version). *Let* $D \subseteq \{0,1\}^n$ *be a non-empty subset of the hypercube (the "domain" on which* f *computes the Hamming weight), and let* $f : \{0,1\}^n \to \{0,\dots,n\}$ *be a function such that* $\forall x \in D, f(x) = |x|$. *Suppose that* $f = h(g_1, \dots, g_m)$ *and that variable* i *is queried by at most* q *of the inner functions* $g_1, \dots, g_m$. *Let* $\boldsymbol{X} \sim D$ *be a random input drawn uniformly over* D. *Then* $\mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \dots, g_m(\boldsymbol{X})] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}$.

We will start by proving a special case to build intuition, then prove the full version using a similar approach.

5.1   Warmup: $q = 1$ for total functions

In preparation for proving Lemma 26 in its full generality, let us prove the special case $q = 1$ of Lemma 22 (the "total functions" version), which is simpler and illustrates the core idea quite well.

▶ **Proposition 29** ("baby version" of Lemma 22). *Suppose that* $\mathrm{HW}_n = h(g_1, \dots, g_m)$ *and that variable* i *is queried by only one of the inner functions* $g_1, \dots, g_m$. *Let* $\boldsymbol{X} \sim \{0,1\}^n$ *be a uniformly random input. Then* $\mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \dots, g_m(\boldsymbol{X})] = 1$. *That is, the outputs of the inner functions* $g_1(\boldsymbol{X}), \dots, g_m(\boldsymbol{X})$ *determine* $\boldsymbol{X}_i$ *completely.*

Proof. Say without loss of generality that the only inner function which queries variable i is g_1. Fix any input x. We will show how to recover x_i from the values $g_1(x), \dots, g_m(x)$.

First, set aside $g_1(x)$, and consider what values $|x|$ could take if we knew only $g_2(x), \dots, g_m(x)$. By our assumption $\mathrm{HW}_n = h(g_1, \dots, g_m)$, $|x| = h(g_1(x), \dots, g_m(x))$, so $|x|$ must belong to the set

$$W = \{h(0, g_2(x), \dots, g_m(x)), h(1, g_2(x), \dots, g_m(x))\}.$$

Now, consider the input $x^{\oplus i}$ (x with its ith coordinate flipped). Since only g_1 depends on the ith coordinate, $x^{\oplus i}$ must share the same output values for $g_2, \dots g_m$. This means that *both* Hamming weights $|x|$ and $|x^{\oplus i}|$ must belong to the set W. But we know that $|x^{\oplus i}|$ is equal to either $|x| + 1$ (when $x_i = 0$) or $|x| - 1$ (when $x_i = 1$), and $|W| \leq 2$, so that means that W must be of the form

$$W = \{|x|, |x^{\oplus i}|\} = \{w, w + 1\}$$

for some w.

So to find the value of x_i, it suffices to check whether $|x| = w$ (in which case $x_i = 0$) or $|x| = w + 1$ (in which case $x_i = 1$). Since both W and $|x| = h(g_1(x), \ldots, g_m(x))$ can easily be computed from $g_1(x), \ldots, g_m(x)$, this gives us a way to recover x_i from $g_1(x), \ldots, g_m(x)$. ◄

5.2 General case

The proof of Lemma 26 is a generalization of the trick above: we fix the outputs of the inner functions that *do not* query variable i, then consider the (relatively small) set of possible values for $|\boldsymbol{X}|$ given those outputs, and finally use our knowledge of $|\boldsymbol{X}|$ to guess $\boldsymbol{X}_i$ better than random chance.

Proof of Lemma 26. First, let us show that the inequality holds when $q = 0$. If $q = 0$, then f does not depend on variable i at all, so for any $x \in \{0,1\}^n$, $f(x) = f(x^{\oplus i})$. This means we cannot simultaneously have $f(x) = |x|$ and $f(x^{\oplus i}) = |x^{\oplus i}|$. Thus whenever $x \in D$, $x^{\oplus i}$ cannot also be in D. As a result, $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] = 1$, so the right hand side becomes

$$1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)} = 1 + 1 - 1 = 1,$$

and the inequality is trivially verified. Having proved the inequality for $q = 0$, we may assume that $q \geq 1$ for the remainder of the proof.

Let $J_i \subseteq [m]$ be the set of inner functions that depend on variable i (so $|J_i| \leq q$), and let $\overline{J_i} := [m] \setminus J_i$. Using the fact that $g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})$ determine $|\boldsymbol{X}|$ completely, we can weaken the conditioning to get

$$\mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \leq \mathbf{H}[\boldsymbol{X}_i \mid \{g_j(\boldsymbol{X})\}_{j \in \overline{J_i}}, |\boldsymbol{X}|],$$

and by definition of conditional entropy,

$$
\begin{aligned}
&\mathbf{H}[\boldsymbol{X}_i \mid \{g_j(\boldsymbol{X})\}_{j \in \overline{J_i}}, |\boldsymbol{X}|] \\
&\quad = \mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[-\log \mathbf{Pr}[\boldsymbol{X}_i = \boldsymbol{Y}_i \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]] \\
&\quad = \mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|])],
\end{aligned}
$$

where $H_2(p) := -p \log p - (1-p) \log(1-p)$ is the binary entropy function. Thus, to prove the lemma, it suffices to show that

$$\mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|])] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}. \tag{5}$$

Partitioning D according to the outputs of the inner functions that do not query the variable i. To prove this inequality, let us first split into cases according to the outputs of the inner functions that do not query variable i. Fix the output values $v \in \{0,1\}^{\overline{J_i}}$ of the inner functions that do not query variable i, and suppose that those output values are achievable by some input in D (that is, there exists some input $x \in D$ such that $\forall j \in \overline{J_i}, g_j(x) = v_j$). Let $S_v := \{x \in \{0,1\}^n \mid \forall j \in \overline{J_i}, g_j(x) = v_j\}$ be the set of inputs that produce these inner function outputs. Note that the sets $D \cap S_v$ form a partition of D. As we will see later, a key property of S_v is that its image under f has cardinality at most 2^q.

Let us call v *good* if $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \mid \boldsymbol{X} \in S_v] \leq 2 \cdot \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D]$ (that is, v is good if $\boldsymbol{X}^{\oplus i}$ is not much more likely to lie outside of D when $\boldsymbol{X}$ is in S_v than on average). By Markov's inequality, the sets S_v with good v account for most of the probability mass: that is, $\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \geq 1/2$.

Our objective will be to show that for every good v,

$$\operatorname*{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}. \quad (6)$$

Informally, the task is: for any good $v \in \{0, 1\}^{\overline{J_i}}$ (representing the outputs of the inner functions that do not query variable i), based only on v and the Hamming weight $|\boldsymbol{X}|$, guess the value of $\boldsymbol{X}_i$ slightly better than random chance.

Let us see why (6) implies (5):

$$\operatorname*{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|])]$$

$$= \sum_v \mathbf{Pr}[\boldsymbol{X} \in S_v] \cdot \operatorname*{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v]$$

(by the law of total expectation)

$$\leq \sum_v \mathbf{Pr}[\boldsymbol{X} \in S_v] \cdot \begin{cases} 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)} & \text{if } v \text{ is good} \\ 1 & \text{otherwise} \end{cases}$$

(by (6) and because $H_2(\cdot) \leq 1$)

$$= 1 + \left(\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \right) \left(\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)} \right)$$

$$\leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - \left(\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \right) 2^{-O(q)}$$

(because $\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \leq 1$)

$$\leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}. \qquad \text{(because } \sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \geq 1/2 \text{ and } q > 0\text{)}$$

Proof overview for (6). Fix some good $v \in \{0, 1\}^{\overline{J_i}}$, and let $W_v \coloneqq \{|x| \mid x \in D \cap S_v\}$ be the set of possible Hamming weights for inputs in $D \cap S_v$. When $x \in D$, we know that $|x| = f(x) = h(g_1(x), \dots, g_m(x))$, and when in addition $x \in S_v$, then for all $j \in \overline{J_i}$, the output value $g_j(x)$ is already fixed to v_j. Therefore, when $x \in D \cap S_v$, the Hamming weight $|x|$ can only depend on the remaining q output values $g_j(x)$ for $j \in J_i$, so there are at most 2^q possible values for $|x|$. In other words, $|W_v| \leq 2^q$.

Informally, using the fact that W_v is small, we will show that there exists a Hamming weight $w^* \in W_v$ that occurs with significant probability within S_v, and such that conditioned on $|\boldsymbol{X}| = w^*$, the probability that $\boldsymbol{X}_i = 1$ is not too close to $1/2$. This means that when $|\boldsymbol{X}| = w^*$ we can guess $\boldsymbol{X}_i$ slightly better than random chance, and even if we just guess randomly when $|\boldsymbol{X}| \neq w^*$, we will have achieved better-than-random-chance accuracy overall. Formally, we will show that $\exists w^* \in W_v$ such that

$$\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid x \in S_v] \geq 2^{-O(q)} \tag{7}$$

and

$$\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*] \leq \frac{1}{2} - 2^{-O(q)}. \tag{8}$$

Let us see why (7) and (8) together imply (6):

$$\mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v]$$

$$= \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v]$$
$$\cdot \mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v \wedge |\boldsymbol{Y}| = w]$$
$$\text{(by the law of total expectation)}$$

$$= \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v]$$
$$\cdot \mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w]) \mid \boldsymbol{Y} \in S_v \wedge |\boldsymbol{Y}| = w]$$
$$\text{(replace } |\boldsymbol{Y}| \text{ by } w \text{ using the conditioning)}$$

$$= \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v] \cdot H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w])$$
$$\text{(the expectation was constant)}$$

$$\leq \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v] \cdot \begin{cases} H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*]) & \text{if } w = w^* \\ 1 & \text{otherwise} \end{cases}$$
$$\text{(because } H_2(\cdot) \leq 1)$$

$$= 1 - \mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] \cdot (1 - H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*]))$$
$$\text{(rearrange and use the fact that probabilities sum to 1)}$$

$$\leq 1 - 2^{-O(q)}\left(1 - H_2\left(1/2 - 2^{-O(q)}\right)\right) \qquad \text{(by (7) and (8))}$$
$$= 1 - 2^{-O(q)}\left(1 - \left(1 - 2^{-O(q)}\right)\right) \qquad \text{(because } H_2(1/2 - p) = 1 - \Omega(p^2))$$
$$= 1 - 2^{-O(q)}.$$

Existence of the weight w^*. Let us now find this magical Hamming weight w^*.

First of all, assume that $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] \leq 2^{-10q}$ (otherwise, we can replace the right-hand side of (6) by 1 and the inequality becomes trivial). Since v is good, this means that $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \mid \boldsymbol{X} \in S_v] \leq 2 \cdot 2^{-10q} = 2^{-10q+1}$.

When $x \in S_v$, $x^{\oplus i}$ must also be in S_v, since flipping variable i does not affect the output of any inner function g_j for $j \in \overline{J_i}$. Informally, this means that for most inputs $x \in D \cap S_v$ (those for which $x^{\oplus i} \in D$), we can pair it up with another input $x^{\oplus i}$ also in $D \cap S_v$. We can identify each such pair by looking at the value of $x^{(i \mapsto 0)}$ (x with its ith coordinate set to 0). Our argument will rely crucially on this pairing of inputs.

For any integer $-1 \leq w \leq n$, let $p_w := \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \wedge |\boldsymbol{X}^{(i \mapsto 0)}| = w \mid \boldsymbol{X} \in S_v]$ (note that $p_{-1} = 0$ somewhat vacuously). It is clear that $p_w = 0$ whenever $w, w + 1 \notin W_v$. This means that the sequence $\{p_w\}_{w \in \{-1,\dots,n\}}$ has at most $2|W_v| \leq 2^{q+1}$ nonzero elements. Moreover $\sum_{w=-1}^{n} p_w = \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \mid \boldsymbol{X} \in S_v] \geq 1 - 2^{-10q+1}$, so we must have $\max_{w=-1}^{n} p_w \geq (1 - 2^{-10q+1})/2^{q+1} \geq 2^{-q-2}$. Now, again using the fact that the sequence $\{p_w\}_{w \in \{-1,\dots,n\}}$ has at most 2^{q+1} nonzero elements, to go from $p_{-1} = 0$ to this maximum value of $\geq 2^{-q-2}$, the sequence must contain some "upwards jump" of at least $2^{-q-2}/2^{q+1} = 2^{-2q-3}$, so there must exist a weight $w^* \in \{0,\dots,n\}$ such that $p_{w^*} \geq p_{w^*-1} + 2^{-2q-3}$.

Let us show that $w^* \in W_v$ and that it satisfies (7) and (8). First, note that the set $\{x \in D \cap S_v \mid |x| = w^*\}$ includes at least the following two disjoint sets:

(i) the inputs $x \in D \cap S_v$ such that $x^{\oplus i} \in D$, $x_i = 0$, and $|x^{(i \mapsto 0)}| = w^*$;

(ii) the inputs $x \in D \cap S_v$ such that $x^{\oplus i} \in D$, $x_i = 1$ and $|x^{(i \mapsto 0)}| = w^* - 1$.

It is easy to see that there are exactly $\frac{p_{w^*}}{2}|D \cap S_v|$ inputs of type (i) and $\frac{p_{w^*-1}}{2}|D \cap S_v|$ inputs of type (ii). This means that

$$\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] \geq \frac{p_{w^*} + p_{w^*-1}}{2} \qquad\qquad (9)$$

$$\geq \frac{p_{w^*}}{2} \qquad\qquad \text{(because } p_{w^*-1} \text{ is a probability)}$$

$$\geq \frac{2^{-2q-3}}{2} \qquad\qquad \text{(because } p_{w^*} \geq p_{w^*-1} + 2^{-2q-3} \geq 2^{-2q-3})$$

$$= 2^{-O(q)},$$

satisfying (7). Also, given that $\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] \geq 2^{-O(q)} > 0$, there must be some $x \in D \cap S_v$ such that $|x| = w^*$, so $w^* \in W_v$. Finally,

$$\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*]$$

$$= \frac{\mathbf{Pr}[\boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}{\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]} \qquad\qquad \text{(by definition of conditional probability)}$$

$$= \frac{\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \wedge \boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \wedge \boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}{\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}$$

$$\text{(split according to } \boldsymbol{X}^{\oplus i} \overset{?}{\in} D)$$

$$\leq \frac{\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \mid \boldsymbol{X} \in S_v] + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \wedge \boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}^{(i \mapsto 0)}| = w^* - 1 \mid \boldsymbol{X} \in S_v]}{\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}$$

$$\text{(logical consequences)}$$

$$\leq \frac{2^{-10q+1} + \frac{p_{w^*-1}}{2}}{\frac{p_{w^*} + p_{w^*-1}}{2}} \qquad\qquad \text{(because there are } \frac{p_{w^*-1}}{2}|D \cap S_v| \text{ inputs of type (ii), and by (9))}$$

$$= \frac{2^{-10q+2} + p_{w^*-1}}{p_{w^*} + p_{w^*-1}}$$

$$\leq \frac{2^{-10q+2} + (p_{w^*} - 2^{-2q-3})}{p_{w^*} + (p_{w^*} - 2^{-2q-3})} \qquad\qquad \text{(because } p_{w^*} \geq 2^{-2q-3} > 2^{10q+2} \text{ and } p_{w^*-1} \leq p_{w^*} - 2^{-2q-3})$$

$$= \frac{1}{2} - \frac{2^{-2q-3}/2 - 2^{-10q+2}}{2p_{w^*} - 2^{-2q-3}}$$

$$\leq \frac{1}{2} - \frac{2^{-2q-5}}{2p_{w^*} - 2^{-2q-3}} \qquad\qquad \text{(because } q \geq 1)$$

$$< \frac{1}{2} - \frac{2^{-2q-5}}{2} \qquad\qquad \text{(because } p_{w^*} \leq 1)$$

$$= \frac{1}{2} - 2^{-O(q)}, \qquad\qquad \text{(because } q \geq 1)$$

thus (8) is satisfied, and this concludes the proof. ◀

6 Conclusion

In this section we outline several interesting directions for future work.

6.1 A plan towards better depth-3 lower bounds

As we pointed out in the introduction, if it were possible to compute MAJ_n as a fan-in-$O(\sqrt{n})$ function of fan-in-$O(\sqrt{n})$ functions, there would be $2^{O(\sqrt{n})}$-size depth-3 circuits for MAJ_n. By showing that $\mathsf{CC}_{\Theta(\sqrt{n})}(\mathrm{MAJ}_n) = \omega(\sqrt{n})$, we ruled out this particular way of obtaining $2^{O(\sqrt{n})}$-size depth-3 circuits for MAJ_n. Theorem 2 is therefore a necessary first step for showing that MAJ_n requires $2^{\omega(\sqrt{n})}$-size depth-3 circuits.

If we keep considering constructions of the form $\mathrm{MAJ}_n = h(g_1, \ldots, g_m)$, there are several ways we can make both the outer function h and the inner functions $g_1, \ldots, g_m$ more powerful while still giving depth-3 circuit upper bounds. For example, one could try to prove that

MAJ_n cannot be computed as a fan-in-$O(\sqrt{n})$ function of depth-$O(\sqrt{n})$ decision trees, or even as a depth-$O(\sqrt{n})$ decision tree of depth-$O(\sqrt{n})$ decision trees. Indeed, if such a construction were possible, the outer function could be transformed into a size-$2^{O(\sqrt{n})}$ DNF (resp. CNF), while the inner functions and their negations could be transformed into size-$2^{O(\sqrt{n})}$ CNFs (resp. DNFs), which would give a $2^{O(\sqrt{n})}$-size Σ^3 (resp. Π^3) circuit for MAJ_n.

This motivates the following plan towards proving better depth-3 lower bounds for the majority function: prove lower bounds against computing MAJ_n as $h(g_1, \ldots, g_m)$, where the inner and outer functions are replaced by increasingly powerful objects. For now, let us conjecture the following first steps, which could be solved independently.

▶ **Conjecture 30** (making the inner functions more powerful). MAJ_n *cannot be represented as* $h(g_1, \ldots, g_m)$, *where* $m = O(\sqrt{n})$ *and each* g_j *is a depth-*$O(\sqrt{n})$ *decision tree.*

▶ **Conjecture 31** (making the outer function more powerful). MAJ_n *cannot be represented as* $h(g_1, \ldots, g_m)$, *where* h *is a depth-*$O(\sqrt{n})$ *decision tree and each* g_j *depends on only* $O(\sqrt{n})$ *variables.*

In fact, we do not have to climb very far up the power ladder in order to reach the full power of depth-3 circuits. Indeed, if we were to replace "depth-$O(\sqrt{n})$ decision tree" by "size-$2^{O(\sqrt{n})}$ DNF" (resp. CNF) in Conjecture 31, this would already be equivalent to the conjecture that MAJ_n cannot be represented by Σ^3 (resp. Π^3) circuits of size $2^{O(\sqrt{n})}$.[8]

6.2 Further applications of our techniques

An information theoretic toolbox for circuit lower bounds?

In this paper, we showed (in Lemma 22 and Lemma 26) how we can trace the information flow within a circuit of arbitrary functions to prove tight lower bounds. Could one prove lower bounds against boolean circuits via the same technique? In particular, it would be interesting to prove a statement analogous to Lemma 22, but quantifying the mutual information between internal nodes of a boolean circuit.

One challenge of accomplishing this is in the difference in power between the inner functions and the AND/OR gates of boolean circuits. Since the boolean gates are so much weaker than the inner functions (which compute arbitrary boolean functions), we would need to prove much stronger statements about the information flow in order to get tight lower bounds. Nonetheless, we believe that challenges like Open Problem 1 require techniques that can precisely identify the weaknesses of small-depth circuits, and that information theory is a well-suited tool for this task.

The usefulness of multi-output functions

Multi-output functions played a key role in two steps of the proof. These steps can be distilled as a general plan for proving lower bounds for single-output functions as follows:

(1) Prove a lower bound for a multi-output function (in our case, HW_n).
(2) Show that the act of computing a desired single-output function (in our case, MAJ_n) essentially entails computing the multi-output function (perhaps on a smaller set of inputs).

---

[8] In fact, this would still be true even if we replaced "each g_j depends only on $O(\sqrt{n})$ variables" by "each g_j is an OR of $O(\sqrt{n})$ variables".

We believe that this general approach could be applied to a much wider range of models. For example, consider the problem of improving the best lower bounds for depth-3 circuits. If one could prove a $2^{\omega(\sqrt{n})}$ lower bound against depth-3 circuits of HW_n, then this would *automatically* give us better lower bounds for an explicit function (we would get a lower bound of $2^{\omega(\sqrt{n})}/\log n = 2^{\omega(\sqrt{n})}$ for at least one of the output bits of HW_n). In all likelihood, this could be extended (perhaps by the approach in step (2)) to work for MAJ_n, thereby resolving Open Problem 1.

We usually think of single-output functions as being components of multi-output functions, but this work shows that multi-output functions can be hidden within single-output functions as well. We hope that this will encourage the incorporation of multi-output functions into the lexicon of hard functions to prove lower bounds for.

Random linear codes

As a concrete example of functions that one might prove strong lower bounds for using our techniques, we propose random linear error-correcting codes.

In a celebrated work, Paturi et al. [23] showed that error-correcting codes[9] with large enough distance and enough codewords require size-$2^{1.282\sqrt{n}}$-size Σ^3 circuits, which remains to date the strongest lower bound proved for any explicit function. A natural question is, was this technique tight? Can we obtain $2^{\omega(\sqrt{n})}$ lower bounds for all "good enough" error-correcting codes?

Unfortunately, the answer is at least partially negative. Leffman, Pudlák, and Savický [19] showed that there exist linear codes of large distance which have *sparse* parity check matrices. More precisely, they show the existence of a linear code f of distance $n^{\Omega(1)}$ whose parity check matrix has $\sqrt{n}$ rows, each of which has only $O(\sqrt{n})$ non-zero entries. This means that we can write $f = h(g_1, \ldots, g_{\sqrt{n}})$, where each of the inner functions g_j computes a parity check over $O(\sqrt{n})$ variables, and h is the AND function. Using the notation of this paper, $\mathsf{CC}_{O(\sqrt{n})}(f) \leq \sqrt{n}$, and thus f has size-$2^{O(\sqrt{n})}$ Σ^3 circuits.

However, this code has small depth-3 circuits precisely because its parity check matrix is sparse. It is easy to see by a counting argument that most linear codes do *not* have sparse parity check matrices. So it is natural to wonder whether a random linear code (a code whose parity check matrix is chosen uniformly at random) might require large Σ^3 circuits. Concretely, we conjecture the following.

▶ **Conjecture 32.** *Let $\ell \ll k \ll n$. Let $H \sim \{0,1\}^{\ell \times n}$ be a random matrix of ℓ parity checks, and let*

$$f : \{0,1\} \to \{0,1\}^n : x \mapsto \begin{cases} 1 & \text{if } Hx = \vec{0} \\ 0 & \text{otherwise} \end{cases}$$

be the corresponding linear code. Then, with high probability,
  (i) $\mathsf{CC}_k(f) \geq \Omega\left(\frac{n}{k} \cdot \frac{\ell}{\log(n/k)}\right)$;
  (ii) *f requires Σ^3 circuits of size $2^{\Omega(\sqrt{n\ell/\log n})}$.*

The proposed bound $\Omega(\frac{n}{k} \cdot \frac{\ell}{\log(n/k)})$ in point (i) comes from the observation that one can verify any $\log(n/k)$ parity check using $O(n/k)$ inner functions: first split the variables into $O(n/k)$ sets according to their coefficients in each of the $\log(n/k)$ parity check, then compute the parity of each of those $O(n/k)$ sets. Point (i) simply conjectures that this observation

---

[9] More precisely, functions computing whether their input x belongs to a fixed error-correcting code.

gives the best upper bound on $\mathsf{CC}_k(f)$. The proposed bound $2^{\Omega(\sqrt{n\ell/\log n})}$ in point (ii) is obtained by conjecturing that the best Σ^3 circuit size for f will be obtained as $2^{O(\mathsf{CC}_k(f)+k)}$ through the connection with composition complexity (see Section 1.2.2), then balancing the sum by setting $k := \sqrt{n\ell/\log n}$.

If Conjecture 32 is true, it would give lower bounds of the form $2^{n^{1-\epsilon}}$ against Σ^3 circuits. It seems likely to us that, depending on how the proof works, the randomness could then be lifted, and it could be extended to some explicit *pseudorandom* linear code.

We think the techniques in our paper would be particularly well-suited to proving Conjecture 32. Indeed, just like majority has a natural multi-output analog (the Hamming weight function), f has an even more obvious corresponding multi-output function: the function $\vec{f}(x) := Hx$ which gives the outputs of all the parity checks. Therefore, it seems plausible that one could first prove a lower bound for $\vec{f}$ using techniques similar to the ones in Section 3, then extend it to f using our framework for bootstrapping lower bounds from multi-output functions to binary-output functions, which we presented in Section 4.

References

1 Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

2 Noga Alon and Wolfgang Maass. Meanders, ramsey theory and lower bounds for branching programs. In *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, pages 410–417. IEEE, 1986.

3 Kazuyuki Amano and Masafumi Yoshida. Depth two (n-2)-majority circuits for n-majority. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 101(9):1543–1545, 2018.

4 László Babai, Pavel Pudlák, Vojtech Rödl, and Endre Szemerédi. Lower bounds to the complexity of symmetric boolean functions. *Theoretical Computer Science*, 74(3):313–323, 1990.

5 Ravi B Boppana. The average sensitivity of bounded-depth circuits. *Information processing letters*, 63(5):257–261, 1997.

6 Christian Engels, Mohit Garg, Kazuhisa Makino, and Anup Rao. On expressing majority as a majority of majorities. *SIAM Journal on Discrete Mathematics*, 34(1):730–741, 2020.

7 Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *Proceedings of the 22nd IEEE Annual Symposium on Foundations of Computer Science*, pages 260–270, 1981.

8 Dmitry Gavinsky, Or Meir, Omri Weinstein, and Avi Wigderson. Toward better formula lower bounds: an information complexity approach to the krw composition conjecture. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 213–222, 2014.

9 Oded Goldreich and Avishay Tal. Matrix rigidity of random toeplitz matrices. *computational complexity*, 27(2):305–350, 2018.

10 Oded Goldreich and Avishay Tal. On constant-depth canonical boolean circuits for computing multilinear functions. In *Computational Complexity and Property Testing*, pages 306–325. Springer, 2020.

11 Oded Goldreich and Avi Wigderson. On the size of depth-three boolean circuits for computing multilinear functions. In *Computational Complexity and Property Testing*, pages 41–86. Springer, 2020.

12 Johan Håstad. *Computational Limitations for Small Depth Circuits*. MIT Press, Cambridge, MA, 1986.

13   Johan Hastad, Stasys Jukna, and Pavel Pudlák. Top-down lower bounds for depth 3 circuits. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 124–129. IEEE, 1993.

14   Pavel Hrubes, Sivaramakrishnan Natarajan Ramamoorthy, Anup Rao, and Amir Yehuday-off. Lower bounds on balancing sets and depth-2 threshold circuits. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

15   Pavel Hrubes and Anup Rao. Circuits with medium fan-in. In *30th Conference on Computational Complexity (CCC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

16   Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 5. Springer, 2012.

17   Maria Klawe, Wolfgang J Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 480–487, 1984.

18   Alexander S Kulikov and Vladimir V Podolskii. Computing majority by constant depth majority circuits with low fan-in gates. *Theory of Computing Systems*, 63(5):956–986, 2019.

19   Hanno Lefmann, Pavel Pudlák, and Petr Savicky. On sparse parity check matrices. *Designs, Codes and Cryptography*, 12(2):107–130, 1997.

20   Or Meir and Avi Wigderson. Prediction from partial information and hindsight, with application to circuit lower bounds. *computational complexity*, 28(2):145–183, 2019.

21   Edward I Nechiporuk. A boolean function. *Engl. transl. in Sov. Phys. Dokl.*, 10:591–593, 1966.

22   Ilan Newman and Avi Wigderson. Lower bounds on formula size of boolean functions using hypergraph entropy. *SIAM Journal on Discrete Mathematics*, 8(4):536–542, 1995.

23   Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.

24   Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 566–574. IEEE, 1997.

25   Gleb Posobin. Computing majority with low-fan-in majority queries. *arXiv preprint*, 2017. `arXiv:1711.10176`.

26   Alexander A Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *International Symposium on Fundamentals of Computation Theory*, pages 47–60. Springer, 1991.

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

28   Benjamin Rossman. The average sensitivity of bounded-depth formulas. *computational complexity*, 27(2):209–223, 2018.

29   Srikanth Srinivasan. Personal communication, 2015.

30   Leslie G Valiant. Exponential lower bounds for restricted monotone circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 110–117, 1983.

31   Guy Wolfovitz. The complexity of depth-3 circuits computing symmetric boolean functions. *Information Processing Letters*, 100(2):41–46, 2006.

32   Andrew Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 1–10, 1985.