On the Computational Power of Radio Channels

Mark Braverman
Princeton University, Princeton, NJ, USA
mbraverm@cs.princeton.edu

Gillat Kol
Princeton University, Princeton, NJ, USA
gillat.kol@gmail.com

Rotem Oshman
Tel Aviv University, Israel
roshman@tau.ac.il

Avishay Tal
UC Berkeley, CA, USA
atal@berkeley.edu

Abstract

Radio networks can be a challenging platform for which to develop distributed algorithms, because the network nodes must contend for a shared channel. In some cases, though, the shared medium is an advantage rather than a disadvantage: for example, many radio network algorithms cleverly use the shared channel to approximate the degree of a node, or estimate the contention. In this paper we ask how far the inherent power of a shared radio channel goes, and whether it can efficiently compute "classicaly hard" functions such as Majority, Approximate Sum, and Parity.

Using techniques from circuit complexity, we show that in many cases, the answer is "no". We show that simple radio channels, such as the beeping model or the channel with collision-detection, can be approximated by a low-degree polynomial, which makes them subject to known lower bounds on functions such as Parity and Majority; we obtain round lower bounds of the form $\Omega(n^{\delta})$ on these functions, for $\delta \in (0,1)$. Next, we use the technique of random restrictions, used to prove AC^0 lower bounds, to prove a tight lower bound of $\Omega(1/\epsilon^2)$ on computing a $(1 \pm \epsilon)$-approximation to the sum of the nodes' inputs. Our techniques are general, and apply to many types of radio channels studied in the literature.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Communication complexity

Keywords and phrases radio channel, lower bounds, approximate majority

Digital Object Identifier 10.4230/LIPIcs.DISC.2019.8

Funding *Mark Braverman*: Research supported in part by NSF Award CCF- 1525342 and the NSF Alan T. Waterman Award, Grant No. 1933331, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.
Gillat Kol: Research supported by an Alfred P. Sloan Fellowship, the National Science Foundation CAREER award CCF-1750443, and by the E. Lawrence Keyes Jr. / Emerson Electric Co. Award.
Rotem Oshman: This work was done in part while the author was visiting the Simons Institute for the Theory of Computing. Research supported by the Israeli Centers of Research Excellence (I-CORE) program (Center No.4/11) and by BSF Grant No. 2014256.
Avishay Tal: This work was done in part while the author was visiting the Simons Institute for the Theory of Computing. Partially supported by a Motwani Postdoctoral Fellowship and by NSF grant CCF-1763311.

1 Introduction

In a radio network, nodes communicate over a shared channel, and must contend with each other for access to the channel. This can make some essential distributed tasks challenging, e.g., even broadcasting one piece of information across the network is highly non-trivial (see [34] for a survey). On the other hand, many works have observed that the shared channel also presents some opportunities: in a radio network, nodes can *use* the fact that they contend for the same channel to quickly elect a leader [21], approximate their degree (e.g., using the famous Decay algorithm [7]), approximate local sums, and even approximate the PageRank [29].

In this paper we ask: *what is the computational power of a shared radio channel*? Can it efficiently compute "classical hard functions", like majority or parity? How well can it approximate functions like threshold or sum? We use techniques from circuit complexity to show that the computation power of a shared radio channel is subject to significant limitations, and prove tight lower bounds for several functions.

Part of our motivation comes from the problem of *computing an approximate sum*, which is a useful building block in many radio network algorithms: it can be used to compute an approximate degree or estimate the contention, and also to compute the PageRank and related problems [29] (also see Subsection 1.2). Approximate sum is also used in interactive compression [8, 28], where we require a very good (sub-constant) approximation error. In [29] it is shown that in any beeping network (not just single-hop networks), all nodes can compute a $(1 \pm \epsilon)$ approximation of the sum of their neighbors' inputs, in $O(\text{polylog}(n)/\epsilon^2)$ rounds. Our results show that this quadratic dependence on $1/\epsilon$ is tight, even for the simple single hop topology (although we do not match the polylogarithmic dependence on n).

The model. We consider n wireless nodes with inputs $x_1, \ldots, x_n$, communicating over a shared channel (i.e., a single-hop radio network), with the goal of computing some function $f(x_1, \ldots, x_n)$. Our techniques are quite general, and can handle many types of channels considered in the literature, with or without collision detection. We can also handle an *additive network* (see, e.g., [14]), where the nodes receive the XOR of the bits sent on the channel. Generally, as long as the contents of the channel in every round can be computed by a simple Boolean circuit over the values broadcast by the nodes in that round, the channel will be amenable to our techniques.

For simplicity, in the body of the paper we mostly focus on the single-hop *beeping model*: in each round, every node decides whether to *beep* or not to beep; if at least one node decided to beep, all nodes hear a beep, and otherwise they hear silence. In other words, the beeping channel computes the Boolean OR of the inputs to the channel (i.e., of the individual nodes' decisions whether to beep). The beeping model, and other related models, have received a lot of attention in recent years [2, 36, 3, 24, 19, 21, 30, 10, 23, 18, *etc.*], as they provide an abstraction capturing the simplest possible communication primitive: a detectable burst of "energy". This abstraction is well suited for describing wireless networks. In addition, one of the main motivations for studying the beeping model is due to its connections to signal-driven biological systems [4, 31]: e.g., cells communicating by secreting proteins and other chemical markers that are diffused and sensed by neighboring cells, or fireflies reacting to flashes of light from nearby fireflies.

1.1 Our Results and Techniques

We describe two approaches for bounding the computation power of a radio channel. Again, we focus here on the beeping channel.

1.1.1 Lower Bounds via Polynomial Approximations

Our first approach shows that a protocol for the beeping channel can be *approximated by a low-degree polynomial* over the inputs, where the degree of the polynomial depends on the number of rounds used by the protocol.

▶ **Theorem 1.1.** *If there is a randomized r-round beeping protocol $\mathcal{P}$ that computes $f : \{0,1\}^n \to \{0,1\}$ with error ϵ, then there is a polynomial $g \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree $O(r^3)$ which agrees with f on all but a 2ϵ-fraction of the inputs in $\{0,1\}^n$.*

Here, $\mathbb{F}_2[X_1, \ldots, X_n]$ denotes the polynomials over $x_1, \ldots, x_n$ with coefficients in $\mathbb{F}_2$.

It is well-known that some functions, such as the parity function, $\textsc{Parity}_n(x_1, \ldots, x_n) = \sum_i x_i \bmod 2$, and the majority function, $\textsc{Majority}_n(x) = [\sum_i x_i \stackrel{?}{\geq} n/2]$, cannot be approximated by a low-degree polynomial, and this allows us to prove round lower bounds in the beeping model.

▶ **Corollary 1.2.** *Any randomized beeping protocol for $\textsc{Parity}_n$ or $\textsc{Majority}_n$ with error $1/10$ requires $\Omega(n^{1/6})$ rounds.*

Along the way, we show that *any deterministic beeping protocol can be simulated by a constant-depth circuit* with unbounded fan-in AND and OR gates. The size of the circuit corresponds to the number of rounds used by the protocol, so we can use known lower bounds for AC^0 to prove deterministic lower bounds in the beeping model. We then use Razborov's Lemma, which shows that an AC^0 circuit can be approximated by a low-degree polynomial [35].

The beeping channel can be replaced by other types of channels: the degree bound of $O(r^3)$ in Theorem 1.1, and consequently the exponent in Corollary 1.2, change depending on the channel. For example, if we use a channel with collision detection, the degree bound changes to r^5 and the corresponding lower bound on $\textsc{Parity}_n$ and $\textsc{Majority}_n$ becomes $\Omega(n^{1/10})$.

1.1.2 Lower Bounds via Random Restrictions

While the technique of approximation by polynomials yields fairly general lower bounds that can be applied to a wide range of functions and models, it falls short of proving tight lower bounds, at least in the case of $\textsc{Majority}_n$ (and also for $\textsc{Parity}_n$ with randomized protocols); this is inherent, because these functions can be computed by polynomials of degree $\sqrt{n}$. We were especially interested in the $\textsc{Majority}$ function, because a lower bound on $\textsc{Majority}$ implies a lower bound on computing a sum; moreover, a lower bound on $\textsc{Approx-Majority}_{n,\epsilon}$, the promise problem of distinguishing whether $\sum_i x_i \geq (1/2 + \epsilon)n$ or whether $\sum_i x_i < (1/2 - \epsilon)n$, implies a lower bound on computing a $(1 \pm \epsilon)$-approximate sum. We were therefore especially motivated to prove tight bounds for $\textsc{Majority}_n$ and $\textsc{Approx-Majority}_{n,\epsilon}$.

To do so, we use another classical technique from circuit complexity, called *random restrictions* (defined in Subsection 4.2). Essentially, we show that for a certain class of functions, there is no "clever" way to use the beeping channel, and we cannot do much better than simply having the nodes speak one after the other. For this simpler setting we can then use known results.

Using this technique, we prove a nearly-tight lower bound on $\textsc{Approx-Majority}_{n,\epsilon}$, and consequently we obtain the same bound on $(1 \pm \epsilon)$-approximate sum. We mention that our proof also yields a similar lower bound for a "weaker" problem, called $\textsc{Coin}_{n,\epsilon}$, see Corollary 4.11.

▶ **Theorem 1.3.** *Every randomized beeping protocol that solves* Approx-Majority$_{n,\epsilon}$ *with error* $1/10$ *and* $\epsilon > c/\sqrt{n}$ *(for some sufficiently large constant* c*), must use* $\Omega\left(\frac{1}{\epsilon^2}\right)$ *rounds.*

The same theorem applies to other "simple" channels, such as the channel with collision detection.

▶ **Corollary 1.4.** *Any randomized beeping protocol for* Majority$_n$ *with error* $1/10$ *requires* $\Omega(n)$ *rounds.*

Our lower bound applies even if nodes have a shared source of randomness. It is easy to see that in this setting, our lower bound is *tight*: sampling $1/\epsilon^2$ random nodes, and having these nodes announce their inputs one after the other, solves Approx-Majority$_{n,\epsilon}$ with constant error. If nodes only have *private* randomness, the problem becomes more challenging, but [29] shows it can still be solved in $O(\log^2/\epsilon^2)$ rounds. Therefore, our result is tight up to the polylogarithmic dependence on n even with private randomness.

We are also able to give a lower bound of $\Omega(1/\epsilon^2)$ on computing a $(1 \pm \epsilon)$-approximation to the size of the network in the beeping model and related models. This problem was studied in [10], where the authors give an algorithm that runs in $O(\log(1/\delta)/\epsilon^2 + \log\log n)$ rounds and succeeds with probability $1 - \delta$, and prove a lower bound of $\Omega(\log(1/\delta)/\epsilon + \log\log n)$. A lower bound of $\Omega(1/(\epsilon^2 \log(1/\epsilon)) + \log\log n)$ is shown in [15] for the related model of *RFID networks*. Computing an approximate sum reduces to the problem of approximating the size of the network, by simply having nodes with input 0 pretend that they are not present; thus, Theorem 1.3 also gives a lower bound of $\Omega(1/\epsilon^2)$ on this problem. The converse is not necessarily true; the lower bounds of [10, 15] rely on being able to choose which nodes participate in the computation, and they do not apply to computing an approximate sum in a fixed-size network. Our result recovers the dependence of $1/\epsilon^2$ in these bounds, and extends it to other models.

1.2 Related Work

The literature on wireless networks is vast; for lack of space, we survey only work that is directly related to out results. Many radio network algorithms use approximate counting as a subroutine, or solve it directly: [25, 13, 26, 27, 32] give constant-factor approximations in polylogarithmic time in the general radio model, under various assumptions, such as whether or not collision detection is present, and whether there is an adversary that can corrupt some messages.

In the beeping model [17], approximate counting was studied in [10], which gives an upper bound of $O(\log\log n + \log(1/\delta)/\epsilon^2)$, where ϵ is the approximation quality and δ is the error probability, and a lower bound of $\Omega(\log\log n + \log(1/\delta)/\epsilon)$. It is assumed in [17] that nodes do not have unique identifiers, and nothing is known a-priori about the size of the network. We show a lower bound of $\Omega(1/\epsilon^2)$ on the same problem, which holds even when nodes do have identifiers, and the size of the network is initially known up to a constant factor. In [29], the *approximate sums* problem is introduced: every node has some number in the range $\{0\} \cup [1, m]$, and each node must compute a $(1 \pm \epsilon)$-approximation to the sum of its neighbors' numbers. The authors give an algorithm for this problem that runs in $O((\log^2 + \log n \log m)/\epsilon^2)$ rounds, and use it to develop algorithms for PageRank and related problems. Our lower bound shows that the quadratic dependence on $1/\epsilon$ of the algorithm from [29] is inherent, even in single-hop networks when the inputs are Boolean, and even when the nodes have public randomness.

In [33], Newport gives a general technique for proving lower bounds in radio networks, and used it to prove and unify existing proofs for the wake-up and broadcast problems in a few different variants of the model (with or without collision detection and multiple channels). The focus in [33] is different than ours: we are mainly interested in the computational power inherent in the broadcast medium, while [33] gives a unified strategy for quantifying the cost of uncertainty and symmetry breaking.

The computational power of the beeping model is also studied in [21], but from a different perspective: [21] characterizes the number of states the nodes must have in order to solve randomized leader election, and show that with this number of states, it is also possible to simulate a logspace Turing machine with a constant number of unary input tapes. In a sense, [21] shows what the beeping model *can* do, while we focus on what it *cannot* do. Unlike [21], we do not restrict the computational power of the nodes themselves; they can have unbounded space or solve undecidable problems if they so wish.

A somewhat related model to radio networks is *RFID networks*, and many approximate counting protocols have been developed for that setting – we refer to [15] for a survey of this line of research. It is shown in [15] that RFID networks require $\Omega(\log\log n + 1/(\epsilon^2 \log(1/\epsilon)))$ to estimate their size to within $(1 \pm \epsilon)$, and this is almost tight.

2 Preliminaries

Notation. We use bold-face letters to denote random variables. All logarithms are base 2 (unless stated otherwise).

Given a string $s = s_1 \dots s_k \in \{0,1\}^*$ and an index $i \le k$, we let $s_{<i} = s_1 \dots s_{i-1}$ denote the length-$(i-1)$ prefix of s (with $s_{<1} = \varepsilon$, the empty string).

2.1 The Beeping Model

Model. In the single hop *beeping model*, a set of n nodes communicate over a shared channel. We assume that each node i has a single input bit, $x_i \in \{0,1\}$.

Communication occurs in synchronous rounds. In every such round, each node can choose to either beep or listen. If a node $i \in [n]$ listens in round m, it can only distinguish between silence (no other node beeps in round m) or the presence of one or more beeps (at least one other node beeps in round m). Thus, we can think of each node as broadcasting a bit in every round (where 0 corresponds to silence and 1 means beeping), and all nodes receive the OR of the n bits sent in this round.

Protocols. An *r-round deterministic protocol* for the beeping model specifies, for each node $i \in [n]$, two functions: a *broadcast function*, specifying whether node i should beep in each round $\ell = 1, \dots, r$, as a function of its input x_i and the transcript (i.e., the contents of the channel) in rounds $1, \dots, \ell - 1$; and an *output function*, which takes the complete r-round transcript and the input x_i of node i, and determines what value node i should output at the end of the protocol. Since we are concerned only with Boolean functions here, we assume for convenience that the protocol's output is the contents of the channel in the last round (i.e., if the transcript is $b_1, \dots, b_r$, then all nodes output b_r). We refer to r as the *length* or *running time* of the protocol.

A *randomized* beeping protocol $\mathcal{P}$ is a distribution over deterministic beeping protocols. Note that this definition corresponds to assuming that the nodes have *shared randomness*, an assumption that is usually avoided when designing beeping protocols, but this only strengthens our lower bounds.

The *length* of a randomized protocol $\mathcal{P}$ is the maximum length of a deterministic protocol in the support of $\mathcal{P}$.

3 Beeping Lower Bounds via Polynomial Approximations

We begin by showing that the beeping model is no more powerful than AC^0 *circuits* where deterministic protocols are concerned, and for randomized protocols, a beeping protocol can be *approximated by a low-degree polynomial*. For both computation models (AC^0 circuits and low-degree polynomials), powerful lower bounds are known, and we can then apply these lower bounds to the beeping model.

3.1 From Deterministic Beeping Protocols to Circuits

Fix an r-round deterministic protocol P for the beeping model. We would like to simulate P by a "simple" circuit C_P whose output agrees with P on all inputs $x \in \{0,1\}^n$.

The natural approach would be to try to simulate P round-by-round: in each round, every node decides whether or not to beep, as a function of the transcript (i.e., the contents of the channel) so far, and the contents of the channel is an OR over the nodes' decisions in the current round. Thus, an r-round deterministic protocol can be represented as a circuit of depth $O(r)$ and size $2^{O(r)}$, where the exponential size comes from the fact that each node's decision whether to beep in a given round can be an arbitrarily complex function of the transcript so far and its input. Unfortunately, no lower bounds are currently known against such circuits, except when r is very small (e.g., less than logarithmic in the input size).

Instead of constructing a circuit with high depth, given a deterministic beeping protocol, we transform it into a circuit of *constant depth*, composed of AND, OR and NOT gates with unbounded fan-in. We can then apply known AC^0 lower bounds against constant-depth circuits.

The circuit we construct is in negation-normal form, i.e., NOT gates are used only on input wires. To simplify the presentation, we assume that the circuit receives as input the n variables $x_1, \ldots, x_n$ on which it computes, as well as their negations, $\overline{x_1}, \ldots, \overline{x_n}$. The *depth* of the circuit is defined to be the maximum number of AND or OR gates on any path from an input (either x_i or $\overline{x_i}$) to the output of the circuit; the *size* of a circuit is the total number of wires.

▶ **Lemma 3.1.** *Let P be a deterministic beeping protocol between n nodes, each node $i \in [n]$ holding an input bit x_i. Assume that P has worst-case running time of r rounds. Then there exists a circuit C_P of depth 3 and size $2^{O(r+\log n)}$ such that $C_P(x) = P(x)$ for all $x \in \{0,1\}^n$.*

Proof. We construct C_P by "flattening" the protocol: instead of simulating it round-by-round, we *guess* an accepting transcript of the protocol (i.e., we guess the contents of the channel in every round, in an execution that leads to output 1), and verify that indeed the protocol would generate this transcript on the input at hand. Here, "guessing" corresponds to an OR over all accepting transcripts; verifying that a given transcript is consistent with the input can be done using a small depth-2 circuit.

Let us describe the construction more formally, from the bottom up.

Observe that for each player i, once we have fixed the transcript $t_{<\ell} \in \{0,1\}^{\ell-1}$ of the rounds preceding round ℓ, player i's decision whether to beep in round ℓ next round depends only on its input x_i. Thus, it is either constant or a literal, x_i or $\overline{x_i}$.

Let $D^i_{t,\ell}$ be the depth-0 circuit representing player i's decision whether to beep or not in round ℓ, when the transcript of the rounds up to ℓ is $t_{<\ell}$. (Again, $D_{t,\ell}$ is either constant or a literal). Our next step is to construct a circuit $R_{t,\ell}$ that "verifies" that round ℓ is consistent with the preceding rounds and the input: that is, $R_{t,\ell}$ outputs 1 on input x iff when P is

executed on input x, if the transcript of the first $\ell - 1$ rounds is $t_{<\ell}$, then that the contents of the channel in round ℓ is indeed t_ℓ. By definition, the contents of the beeping channel is a disjunction over the players' decisions, so we define:

$$R_{t,\ell} = \begin{cases} \bigvee_{i\in[n]} D^i_{t,\ell}, & \text{if } t_\ell = 1, \\ \bigwedge_{i\in[n]} \neg D^i_{t,\ell}, & \text{if } t_\ell = 0. \end{cases}$$

Note that in case $t_\ell = 0$, we want to verify that no player decided to beep, and since the inputs to the circuit are $x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$, we can still verify this using a single AND gate. As a result, $R_{t,\ell}$ is always a single gate; it has depth 1 and size at most n.

Now, let

$$V_t = \bigwedge_{\ell=1}^{r} R_{t,\ell}$$

be a depth-2 circuit of size $O(n \cdot r)$ that checks whether the transcript t would indeed be generated on the current input and outputs 1 iff this is the case.

Finally, let $\mathcal{T} \subseteq \{0,1\}^r$ be the set of accepting transcripts of P (i.e., transcripts that cause the nodes to output 1). The circuit C_P is given by

$$C_P = \bigvee_{t\in\mathcal{T}} V_t,$$

a depth-3 circuit of size $O(n \cdot r \cdot 2^r) = 2^{O(r+\log n)}$, since $|\mathcal{T}| \leq 2^r$ and the size of each sub-circuit V_t is $O(n \cdot r)$. ◀

Using this transformation, we can immediately "import" known lower bounds for AC^0, and obtain, for example, lower bounds for the round complexity of beeping protocols for the Parity and Majority functions.

▶ **Corollary 3.2.** *Any deterministic beeping protocol that computes* $\textsc{Parity}_n$ *or* $\textsc{Majority}_n$ *requires* $\Omega(\sqrt{n})$ *rounds.*

Proof. It is known that any depth-3 circuit for $\textsc{Parity}_n$ or $\textsc{Majority}_n$ must have size $2^{\Omega(\sqrt{n})}$ [22]. Together with Lemma 3.1, we obtain the corresponding lower bounds for the beeping model. ◀

We remark that while every r-round beeping protocol can be simulated by an AC^0 circuit of depth 3 and size $2^{O(r+\log n)}$ (Lemma 3.1), the converse is not true: AC^0 circuits have inherent parallelism, which makes them more powerful than the beeping model. One example is the majority function, which can be computed by an AC^0 circuit of depth 3 and size $2^{O(\sqrt{n})}$, but requires $\Omega(n)$ rounds in the beeping model (as we show in the next section). An example that seems even worse is the function

$$\textsc{Tribes}_{k,\ell}(x_1, \ldots, x_{k\cdot\ell}) = \bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{\ell} x_{i,j} \right).$$

For any $k, \ell \in \mathbb{N}$, the function $\textsc{Tribes}_{k,\ell}$ is computed by an AC^0 circuit of depth 2 and size $k \cdot \ell$. However, it seems plausible that the beeping model requires $\Omega(k)$ rounds to solve $\textsc{Tribes}_{k,\ell}$: even though each inner conjunction can be computed in a single round, we still need to compute k such conjunctions, and it seems this should require $\Omega(k)$ rounds (but we have not proven this intuition).

3.2 Randomized Beeping Protocols

The lower bound technique above applies only to deterministic protocols, since it relies on deterministic lower bounds for AC^0. To extend this approach to randomized protocols, we take it one step further, and use the fact that a small low-depth circuit can be *approximated by a low-degree polynomial*, while "complex" functions like Parity and Majority cannot be approximated by a low-degree polynomial. This is one central approach used to prove AC^0 lower bounds, e.g., [35, 38], and we use the construction from [35]:

▶ **Lemma 3.3** (Razborov's Lemma, [35]). *Given a circuit C of size s and depth d, for any sufficiently small $\epsilon \in (0, 1/2)$, there exists a distribution $\mathcal{G}$ over multivariate polynomials $g(x_1, \ldots, x_n) \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree at most $(\log(s/\epsilon))^d$, such that for all $x \in \{0,1\}^n$,*

$$\Pr_{\boldsymbol{g} \sim \mathcal{G}} [\boldsymbol{g}(x) \neq C(x)] \leq \epsilon.$$

Here, $\mathbb{F}_2[X_1, \ldots, X_n]$ denotes the polynomials over $x_1, \ldots, x_n$ with coefficients in $\mathbb{F}_2$.

Combining Lemma 3.3 with Lemma 3.1 yields the following corollary:

▶ **Corollary 3.4** (Theorem 1.1 restated). *If there is a randomized r-round beeping protocol $\mathcal{P}$ that computes $f : \{0,1\}^n \to \{0,1\}$ with error ϵ, then there is a polynomial $g \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree $O((r + \log n)^3)$ which agrees with f on all but a 2ϵ-fraction of the inputs in $\{0,1\}^n$.*

Now we can rely on the fact that Majority and Parity do not have low-degree approximating polynomials:

▶ **Theorem 3.5** ([35, 38]). *For any polynomial $p \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree t,*

$$\Pr_{\boldsymbol{x} \sim \mathsf{U}(\{0,1\}^n)} [p(\boldsymbol{x}) = \textsc{Majority}_n(\boldsymbol{x})] \leq \frac{1}{2} + O\left(t/\sqrt{n}\right),$$

and similarly for $\textsc{Parity}_n$.

In other words, to obtain *constant* error on the Majority or Parity function, our polynomial must have degree $t = \Omega(\sqrt{n})$. Therefore:

▶ **Corollary 3.6** (Corollary 1.2 restated). *Any randomized beeping protocol for $\textsc{Parity}_n$ or $\textsc{Majority}_n$ with error $1/10$ requires $\Omega(n^{1/6})$ rounds.*

Proof of Corollary 1.2. Given $\mathcal{P}$, we construct for each deterministic protocol P in the support of $\mathcal{P}$ the corresponding circuit C_P from Lemma 3.1, which agrees with P on all inputs. Each C_P has depth 3 and size at most $2^{O(r+\log n)}$, so using Razborov's Lemma (Lemma 3.3), there is a distribution $\mathcal{G}_P$ on polynomials of degree at most

$$\left(\log \frac{2^{O(r+\log n)}}{\epsilon}\right)^3 = O((r + \log n)^3),$$

such that for all $x \in \{0,1\}^n$,

$$\Pr_{\boldsymbol{g} \sim \mathcal{G}_P} [\boldsymbol{g}(x) \neq P(x)] \leq \epsilon.$$

Now let $\mathcal{G}$ be the distribution over $\mathbb{F}_2[X_1, \ldots, X_n]$ where we first pick $\boldsymbol{P} \sim \mathcal{P}$, and then sample $\boldsymbol{g} \sim \mathcal{G}_{\boldsymbol{P}}$. Then, for any $x \in \{0,1\}^n$,

$$\begin{aligned}\Pr_{\boldsymbol{g} \sim \mathcal{G}} [\boldsymbol{g}(x) \neq f(x)] &\leq \Pr_{\boldsymbol{P} \sim \mathcal{P}, \boldsymbol{g} \sim \mathcal{G}_{\boldsymbol{P}}} [\boldsymbol{P}(x) \neq \boldsymbol{g}(x)] + \Pr_{\boldsymbol{P} \sim \mathcal{P}} [\boldsymbol{P}(x) \neq f(x)] \\ &\leq \epsilon + \epsilon = 2\epsilon.\end{aligned}$$

Viewed another way: when we sample $\boldsymbol{x}$ uniformly at random from $\{0,1\}^n$,

$$\mathop{\mathrm{E}}_{\boldsymbol{g}\sim\mathcal{G}}\left[\Pr_{\boldsymbol{x}\sim\mathsf{U}(\{0,1\}^n)}[\boldsymbol{g}(\boldsymbol{x})\neq f(\boldsymbol{x})]\right]\leq 2\epsilon.$$

Therefore, there exists at least one polynomial g in the support of $\mathcal{G}$ such that

$$\Pr_{\boldsymbol{x}\sim\mathsf{U}(\{0,1\}^n)}[g(\boldsymbol{x})\neq f(\boldsymbol{x})]\leq 2\epsilon,$$

and this polynomial, like all polynomials in the support of $\mathcal{G}$, has degree $O((r+\log n)^3)$. ◀

Unfortunately, this is more or less as far as we can go using this approach: there exists a circuit of depth 3 size $2^{O(\sqrt{n})}$ for MAJORITY, and if we relax the requirement to computing MAJORITY correctly on a $(1-\epsilon)$-fraction of inputs (instead of all inputs), the circuit size reduces to $2^{O(n^{1/4})}$ [6]. Therefore, our approach cannot yield a lower bound better than $\Omega(\sqrt{n})$ for deterministic beeping protocols, or better than $\Omega(n^{1/4})$ for randomized protocols, even if we somehow improved the degree of the approximating polynomial in our construction. In the next section, we show that by applying the technique of *random restrictions* directly to a beeping protocol, we can obtain a lower bound of $\tilde{\Omega}(n)$ for $\textsc{Majority}_n$ (and for $\textsc{Parity}_n$), and we can also handle $\textsc{Approx-Majority}_{n,\epsilon}$.

4 Beeping Lower Bounds via Random Restrictions

In this section we use another central technique from circuit complexity, called *random restrictions*, to show that the beeping model cannot compute certain functions efficiently. For lack of space, some proofs are omitted.

Random restriction were used in the seminal proof that AC^0 circuits cannot compute the parity function [5, 20, 40, 22], and since then have found many applications. The basic idea is the following: we are given a Boolean function $f:\{0,1\}^n\to\{0,1\}$, and a circuit (or decision tree) C that computes f. We would like to show that C must have *high depth*. To do this, we randomly choose a subset of the input variables, and fix their values to 0 or 1 at random; this is referred to as "hitting the circuit with a random restriction". We say that an input variable "survived" the restriction if it was not fixed. Then we show:

(a) The complexity of the circuit C is significantly reduced. For example, after hitting an AC^0 circuit with a random restriction where every variable survives with some inverse polynomial probability, the circuit becomes *constant* with high probability.

(b) The complexity of the function f is *not* significantly reduced. For example, hitting the parity function with a random restriction yields a parity over the set of inputs we did not fix.

We begin by describing OR decision trees, a convenient way to represent beeping protocols. Then we formally define random restrictions and analyze what happens to an OR decision tree when hit with a random restriction; finally, we use this machinery to prove a lower bound for approximate majority.

4.1 OR Decision Trees

When each node has a single-bit input $x_i\in\{0,1\}$, a deterministic beeping protocol can be modeled as an *OR decision tree*:

▶ **Definition 4.1** (OR decision tree). *An* OR decision tree of depth d *on variables* $x_1, \ldots, x_n$ *is a binary tree* T *of depth* d*, where each inner node* $v \in \{0,1\}^*$ *is labeled with a disjunction* $D_v = \bigvee_{i=1}^{w_i} \ell_v^i$*. Here, each* ℓ_v^i *is a literal,* $\ell_v^i \in \{x_i, \overline{x_i} : i \in [n]\} \cup \{0,1\}$*. The leafs of the tree are labeled with Boolean values.*

The value *of* T *on an input* $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ *is defined by induction on the height of the tree: the value of a leaf is its label; the value of an inner node* v *is the value of its left subtree if* $D_v(x) = 0$*, or the value of its right subtree if* $D_v(x) = 1$*.*

OR decision trees are a generalization of *decision trees*, where every node queries a single variable (see [12] for a survey). Other generalizations have been considered, *e.g., Parity Decision Trees* (see [41] for a survey). To our knowledge, however, OR decision trees have not been studied before in related contexts.

Beeping protocols as OR trees. To represent a deterministic beeping protocol P as an OR decision tree, we construct the following tree: at each node $v \in \{0,1\}^*$, we place the disjunction $D_v = \bigvee_{i=1}^{n} \ell_v^i$, where $\ell_v^i = P_i(v)$ is 1 if node i beeps after witnessing the transcript v, and 0 otherwise. At each leaf we write the output of P on the corresponding transcript.

A *randomized* beeping protocol is simply a distribution over OR decision trees.

Observe that the depth of the OR decision tree representing a beeping protocol is the number of rounds used by the protocol. Therefore, to prove that a function f does not have a deterministic beeping protocol using r rounds, it suffices to show that every OR decision tree that computes f has depth greater than r, and similarly, for randomized protocols, we must show that every distribution over OR decision trees that computes f with low error has some tree in its support with depth greater than r.

In the sequel, we restrict attention to OR decision trees where at each node of the tree, no variable is queried more than once; that is, the tree does not contain a disjunction of the form $x_i \vee x_i \vee \ldots$ or of the form $x_i \vee \overline{x_i} \vee \ldots$. This is true of trees generated from beeping protocols, but we can also assume it, without loss of generality, about general trees – in the first case we can simply get rid of duplicates, and in the second case the value of the node is always 1, and we can remove it from the tree and replace it with its right subtree.

4.2 Random Restrictions

Let us review the formal definition of a random restriction, and study what happens to an OR decision tree when we hit it with a random restriction.

▶ **Definition 4.2** (Restriction). *An* n*-bit* restriction *is a mapping* $r : \{x_1, \ldots, x_n\} \to \{0, 1, *\}$*. Given an input* $x \in \{0,1\}^n$*, the* restriction *of* x *to* r*, denoted* $x|_r \in \{0,1\}^n$*, is defined by*

$$(x|_r)_i = \begin{cases} x_i & \text{if } r(x_i) = *, \\ r(x_i) & \text{otherwise.} \end{cases}$$

Given a function $f : \{0,1\}^n \to \{0,1\}$*, the* restriction of f to r*, denoted* $f|_r : \{0,1\}^n \to \{0,1\}$*, is defined by* $f|_r(x) = f(x|_r)$*.*

For a parameter $\alpha \in [0,1]$, let $\mathcal{D}_{n,\alpha}$ be the distribution over n-bit restrictions obtained by setting, independently for each $i \in [n]$,

$$r(x_i) = \begin{cases} * & \text{w.p. } 1-\alpha, \\ 0 & \text{w.p. } \alpha/2, \\ 1 & \text{w.p. } \alpha/2. \end{cases}$$

In other words, each input variable survives with probability $1-\alpha$, and otherwise it is fixed to 0 or 1 with equal probability.

What happens to an OR decision tree when we hit it with a random restriction? We show that for any path of length d, with high probability, the *total* number of variables queried along the path drops to $\tilde{O}(d)$. (Recall that even a single node of an OR tree may be labeled by a disjunction of all the variables, so this is a significant.) To show this, let us first formally define the "total cost" of a path, and then study in turn what happens to a disjunction, to a path, and to the entire tree when we hit them with a random restriction.

The total cost of a path. Let π be a path – a sequence of disjunctions, $\pi = D_1, \ldots, D_d$, where $D_i = \bigvee_{j=1}^{w_i} \ell_i^j$ for each i, and each ℓ_i^j is a literal. The *cost* of each disjunction D_i is its width, $\mathrm{cost}(D_i) = w_i$. Define the *total cost* of π as

$$\mathrm{cost}(\pi) = \sum_{i=1}^{d} \mathrm{cost}(D_i).$$

The number of variables queried along π is *at most* $\mathrm{cost}(\pi)$, although it could be smaller, if the same variable is queried by more than one disjunction along π.

Hitting a disjunction with a random restriction. When we hit a disjunction $D = \bigvee_{i=1}^{w} \ell_i$ of width w with a restriction r, the resulting function is also a disjunction: if there is some literal ℓ_i in D that is fixed to 1 (i.e., $\ell_i|_r = 1$), then the value of D becomes fixed, $D|_r = 1$. Otherwise, $D|_r = \bigvee_{i \in S_r} \ell_i$, where $S_r = \{j \in [w] : \ell_j|_r = \ell_j\}$. In this case we say that D *survives* r.

We have

$$\Pr_{\boldsymbol{r} \sim \mathcal{D}_{n,\alpha}} [D \text{ survives } \boldsymbol{r}] = \left(1 - \frac{\alpha}{2}\right)^w,$$

so wide disjunctions have low survival probability. (Recall that we assumed D does not query the same variable more than once, and therefore, the value of each literal after hitting it with r is independent of the other literals.)

Hitting a path with a random restriction. Consider a path $\pi = D_1, \ldots, D_d$. Given a restriction r, let $\pi|_r = D_1|_r, \ldots, D_d|_r$ be the path obtained by hitting each node (disjunction) of π with r.

When we apply a random restriction to π, it has the effect of "killing off" wide disjunctions, and therefore, we expect the total cost of the path to be fairly small:

▶ **Lemma 4.3.** *For any path $\pi = D_1, \ldots, D_d$,*

$$\mathop{\mathrm{E}}_{\boldsymbol{r} \sim \mathcal{D}_{n,1/2}} [\mathrm{cost}(\pi|_{\boldsymbol{r}})] \leq 2|\pi|.$$

Proof. Let w_i be the width of D_i for $i \in [d]$. Since a disjunction of width w survives with probability $(3/4)^w$, the expected cost of $\pi|_{\boldsymbol{r}}$ is

$$\begin{aligned}\mathop{\mathrm{E}}_{\boldsymbol{r}\sim\mathcal{D}_{n,1/2}}[\mathrm{cost}(\pi|_{\boldsymbol{r}})] &= \sum_{i=1}^{|\pi|} \mathop{\mathrm{E}}_{\boldsymbol{r}\sim\mathcal{D}_{n,1/2}}[\mathrm{cost}(D_i|_{\boldsymbol{r}})] \\ &< \sum_{i=1}^{|\pi|} \left(w_i \cdot \Pr_{\boldsymbol{r}\sim\mathcal{D}_{n,1/2}}[D_i \text{ survives } \boldsymbol{r}] \right) \\ &= \sum_{i=1}^{|\pi|} \left(w_i \left(\frac{3}{4}\right)^{w_i} \right) < 2|\pi|.\end{aligned}$$

In the last step we used the fact that $(3/4)^x \cdot x < 2$ for all $x \geq 1$. ◀

Hitting the entire tree with a random restriction. For an OR decision tree T and an input $x \in \{0,1\}^n$, let $\pi_T(x)$ be the computation path of T on x. We let $T|_r$ denote the tree obtained from T by replacing each disjunction D_v with the disjunction $D_v|_r$. Observe that if $f : \{0,1\}^n \to \{0,1\}$ is the function computed by T and r is a restriction, then $f|_r$ is the function computed by $T|_r$.

As we saw above, hitting T with a random restriction has the effect of reducing the cost of each path, with high probability. However, some small fraction of paths may retain high total cost (in fact, this is likely). We deal with these paths by *truncating* them.

▶ **Definition 4.4.** *Given a restriction r, an OR decision tree T and a cost bound $c \in \mathbb{N}$, we define a* truncated *OR decision tree $T|_{r,c}$ as follows: $T|_{r,c}$ is the same as $T|_r$, except that for any node v at depth c, if v is not a leaf, we replace v with a leaf labeled with 0.*

Truncating an OR decision tree can increase its error, but if we choose the depth bound appropriately, the error does not increase by much:

▷ Claim 4.5. Let T be an OR decision tree of depth d on n inputs. Let $\eta \in (0, 1/2)$, and let $\gamma = 2/\eta$. Then for any input $x \in \{0,1\}^n$, we have

$$\Pr_{\boldsymbol{r}\sim\mathcal{D}_{n,1/2}}[T|_{\boldsymbol{r},\gamma d}(x|_{\boldsymbol{r}}) \neq T|_{\boldsymbol{r}}(x|_{\boldsymbol{r}})] \leq \eta.$$

Proof. Let π be the computation path of T on x. By Lemma 4.3 and Markov,

$$\Pr_{\boldsymbol{r}\sim\mathcal{D}_{n,1/2}}[\mathrm{cost}(\pi|_{\boldsymbol{r}}) > (2/\eta)|\pi|] < \eta.$$

Thus, the probability that π needs to be truncated in $T|_{\boldsymbol{r},\gamma d}$ is at most η. If π is not truncated, then it is the same as in $T|_{\boldsymbol{r}}$, and in particular it returns the same answer as $T|_{\boldsymbol{r}}$. Otherwise, we may err, but this happens with probability at most η. ◁

Finally, we observe that an OR decision tree with *low total cost* can be "unrolled" into a *plain* decision tree of *low depth*: we can replace every disjunction of width w by a plain decision tree of depth w that queries the same variables and computes their OR. (In general, any function on w variables can be computed by a decision tree of depth w.)

▷ Claim 4.6. Let T be an OR decision tree where all paths have total cost at most d. Then there exists a (plain) decision tree of depth d that computes the same function as T.

It is known that plain decision trees require high depth to compute an approximate majority; next, we show how we can apply this result to obtain a lower bound for the beeping model.

4.3 The Coin Problem

The *coin problem* [37, 1, 11, 39, 16] is essentially a randomized version of *approximate majority*, and it is often used to prove lower bounds on approximate majority.

In the coin problem, $\textsc{Coin}_{n,\epsilon}$, we have an ϵ-biased coin, where either "heads" or "tails" has probability $1/2+\epsilon$, but we do not know which. The coin is flipped n independent times, and given the n outcomes we need to decide if the coin is biased towards heads or tails. More formally:

▶ **Definition 4.7** (The coin problem). *We say that a function $f:\{0,1\}^n \to \{0,1\}$* solves the coin problem $\textsc{Coin}_{n,\epsilon}$ *with error δ if, for any $a \in \{0,1\}$,*

$$\Pr_{\boldsymbol{x}\sim \mathsf{B}(1/2+(-1)^a\cdot\epsilon)^n}[f(\boldsymbol{x})=a] > 1-\delta.$$

We say that a *distribution* $\mathcal{F}$ over functions $f:\{0,1\}^n \to \{0,1\}$ solves $\textsc{Coin}_{n,\epsilon}$ with error δ if

$$\Pr_{\boldsymbol{x}\sim \mathsf{B}(1/2+(-1)^a\cdot\epsilon)^n,\boldsymbol{f}\sim\mathcal{F}}[\boldsymbol{f}(\boldsymbol{x})=a] > 1-\delta.$$

Abusing the terminology slightly, we will also say that a distribution $\mathcal{T}$ over OR decision trees solves $\textsc{Coin}_{n,\epsilon}$ if the distribution of *functions computed by trees sampled from* $\mathcal{T}$ solves $\textsc{Coin}_{n,\epsilon}$.

It is not hard to see that the best strategy for solving the coin problem is to output the *majority* of the n inputs. By Chernoff, this strategy works as long as $\epsilon > c/\sqrt{n}$, where $c = c(\delta)$ depends on the desired error probability δ. (If $\epsilon \ll c/\sqrt{n}$, it is impossible to solve $\textsc{Coin}_{n,\epsilon}$ with error δ.) Moreover, when $\epsilon > c/\sqrt{n}$ for some sufficiently large $c = c(\delta)$, it suffices to compute an $\epsilon/2$-*approximate majority*: the probability that $|\sum_i \boldsymbol{x}_i - n/2| \le \epsilon/2$ is small, so even an $\epsilon/2$-approximate majority will yield the right answer w.h.p. Thus, a lower bound on solving $\textsc{Coin}_{n,\epsilon}$ immediately yields a lower bound on computing an $\epsilon/2$-approximate majority on n bits (with slightly higher error).

It is known that the coin problem is very difficult for regular decision trees:

▶ **Theorem 4.8** ([16]). *There is a constant $c > 0$ such that for any $\epsilon > c/\sqrt{n}$, the depth of any decision tree that solves $\textsc{Coin}_{n,\epsilon}$ with error $1/10$ is $\Omega(1/\epsilon^2)$.*

Next, we will show that this is also true for OR decision trees (up to constants).

A key property of the coin problem is that when we hit it with a random restriction, it does not become much easier: essentially, fixing a random subset of the outcomes of the coin to 0 or 1 corresponds to *increasing the bias of the coin*, but not by much. This property is used to prove lower bounds for AC^0 using the coin problem.

▶ **Lemma 4.9** ([11, 16]). *Let $\alpha \in [0,1)$. Suppose that $f:\{0,1\}^n \to \{0,1\}$ solves the coin problem $\textsc{Coin}_{n,\epsilon}$ with error δ. The distribution $f|_{\boldsymbol{r}}$ over functions, where $\boldsymbol{r} \sim \mathcal{D}_{n,\alpha}$, solves $\textsc{Coin}_{n,\frac{\epsilon}{1-\alpha}}$ with error δ.*

We showed that when hit with a random restriction, an OR decision tree of depth d "collapses" to a plain decision tree of roughly the same depth. The coin problem, on the other hand, does not become significantly easier when hit with a random restriction (Lemma 4.9). Combining everything, we have:

▶ **Theorem 4.10.** *Let $\mathcal{T}$ be a distribution over OR decision trees that solves $\textsc{Coin}_{n,\epsilon}$ with error at most $1/10$ and $\epsilon > c/\sqrt{n}$ (for some large enough constant c). Then, there is a tree T in the support of $\mathcal{T}$ that has depth $\Omega(1/\epsilon^2)$.*

As we showed in Section 4.1, a beeping protocol can be modeled as an OR decision tree, whose depth corresponds to the number of rounds of the protocol. We therefore obtain the following corollary, which implies Theorem 1.3:

▶ **Corollary 4.11.** *Any randomized beeping protocol that solves* $\textsc{Coin}_{n,\epsilon}$ *with error* $1/10$ *and* $\epsilon > c/\sqrt{n}$ *(for some large enough constant* c*), has* $\Omega(1/\epsilon^2)$ *rounds.*

5 Extension to Other Channel Types

In addition to the beeping model, our techniques can be used to prove lower bounds for other types of wireless channels.

Throughout the paper we have simulated a round of a beeping protocol by a circuit of depth one consisting of a single OR gate. When considering a different wireless channel $\mathcal{C}$, as long as this channel is not too complex, we can also model it as a "simple" circuit C. Then,

- If C is a relatively shallow circuit, then the techniques of Section 3 can be generalized to $\mathcal{C}$, by replacing the OR gates used to simulate one round of the beeping protocol by C-type circuits (see, e.g., Lemma 3.1).
- If hitting C with a random restriction (say, with constant survival probability) yields a circuit that depends on only a small number of inputs w.h.p. then the techniques of Section 4 can also be generalized to $\mathcal{C}$, by replacing the OR trees used to model a beeping protocol by C-trees (trees where each node is labeled by a C-type circuit).

The collision detection channel. Consider, for example, a wireless channel $\mathcal{C}$ with *collision detection*. The output of the channel is one of four symbols, $\perp, \top, 0$ and 1: if no node decides to broadcast, all nodes receive $\perp$; if more than one node decides to broadcast, all nodes receive $\top$; and if exactly one node decides to broadcast, all nodes receive the message it sent.

Next, we implement the operation of $\mathcal{C}$ as a simple circuit C. Note that unlike the beeping channel, in the collision detection channel, each node has two decisions to make: first, whether to broadcast; and second, what value to broadcast (if broadcasting). Accordingly, our circuit C will have inputs $x_1, \ldots, x_n, b_1, \ldots, b_n$, where x_i indicates whether node i broadcast ($x_i = 1$ means that node i did broadcast), and b_i gives the bit broadcast by node i in the case that it did broadcast. The circuit C outputs three bits s, c and b. If $s = 1$, then this is a silent round, i.e., $\mathcal{C}$ outputs $\perp$. If $c = 1$, then a collision has occurred, i.e., $\mathcal{C}$ outputs $\top$. Otherwise, if $s = c = 0$, then the output of $\mathcal{C}$ is the bit b. (We make sure that only one of these three cases holds.)

The circuit C consists of three parts, where each part computes one of c, s and b:

Computing s: the circuit C computes s by taking $s = \bigwedge_{i=1}^{n} \neg x_i$.
Computing c: the circuit C computes c by taking $c = \bigvee_{i \neq j} (x_i \wedge x_j)$.
Computing b: the circuit C computes b by taking $b = \bigvee_{i=1}^{n} (b_i \wedge x_i)$. Note that when $s = c = 0$, there is exactly one node i with $x_i = 1$, and in this case $b_i = \bigvee_{i=1}^{n} (b_i \wedge x_i)$.

The circuit C we constructed is of depth 2 and size $O(n^2)$. We could use it to construct an AC^0 circuit as we did in Lemma 3.1, and the resulting circuit would have depth 4 and size $2^{O(r+\log n)}$. However, we can do better using random restrictions. Instead of explicitly computing the survival probability of each node in the tree, we can appeal to Håstad's Switching Lemma, which analyzes the behavior of DNFs under random restrictions.

A *DNF of width* w is a formula of the form $\bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{k_i} \ell_{i,j} \right)$, where each $\ell_{i,j}$ is a literal, and $k_i \leq w$ for each $1 \leq i \leq m$. Let $\mathrm{DT}(f)$ denote the minimum depth of a (plain) decision tree that computes f.

▶ **Lemma 5.1** (Håstad's Switching Lemma [22]). *Let f be a DNF of width w over n variables, and let $\alpha \leq 1/5$. Then for any $d \geq 0$,*

$$\Pr_{\boldsymbol{r}\sim\mathcal{D}_\alpha}[\mathrm{DT}(f|_{\boldsymbol{r}}) \geq d] \leq (5\alpha w)^d.$$

In particular, whenever $5\alpha w \leq 1/2$, we have $\mathrm{E}_{\boldsymbol{r}\sim\mathcal{D}_\alpha}[\mathrm{DT}(f|_{\boldsymbol{r}})] \leq \sum_{d=0}^{\infty}(5\alpha w)^d \leq 2$. The same holds for CNFs (circuits of the form $\bigwedge_{i=1}^{m}\bigvee_{j=1}^{k_i}\ell_{i,j}$).

In the circuit C that describes the behavior of the collision detection channel, each of the three parts is a DNF or a CNF of width ≤ 2. The Switching Lemma shows that if we hit a C-decision tree with a random restriction $\boldsymbol{r}$ where every variable survives with sufficiently small constant probability (e.g., 1/100), and then "unroll" each C-type node into a decision tree with the smallest depth possible, then for each path π in the original C-decision tree, the expected length of the "unrolled" $\pi|_{\boldsymbol{r}}$ is $O(|\pi|)$. From here, we can proceed exactly as in Section 4, and obtain that the channel with collision detection also requires $\Omega(1/\epsilon^2)$ rounds to solve ϵ-approximate majority.

The additive channel. Another interesting example is the additive channel [14], where in every round, each of the n nodes broadcasts a bit, and each node hears the XOR (or, Parity) of the broadcast bits.

It is known that the Parity function is not very useful when it comes to computing Majority: for example, any depth-3 AC^0 circuit that is additionally equipped with unbounded fan-in Parity gates must still have size at least $2^{\Omega(n^{1/4})}$ to compute $\mathrm{Majority}_n$ [38]. Thus, following the outline from Section 3, we see that a deterministic protocol for the additive single-hop network requires $\Omega(n^{1/4})$ rounds to compute $\mathrm{Majority}_n$.

Similarly to our approach in Section 4, one can model protocols over the additive channel as *parity decision trees*, objects that have been extensively studied (see [41] for a survey). Known lower bounds on the depth of a parity decision tree that computes a specific function, imply the same lower bound in the additive channel model (e.g., the lower bound on the parity decision tree complexity of the recursive majority function in [9]).

References

1 Scott Aaronson. BQP and the polynomial hierarchy. In *Symposium on Theory of Computing (STOC)*, pages 141–150, 2010.

2 Yehuda Afek, Noga Alon, and Ziv Bar-Joseph. Beeping an MIS. *Manuscript*, 2011.

3 Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. Beeping a maximal independent set. *Distributed Computing*, 26(4):195–208, 2013.

4 Yehuda Afek, Noga Alon, Omer Barad, Eran Hornstein, Naama Barkai, and Ziv Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science*, 331(6014):183–185, 2011.

5 Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of pure and applied logic*, 24(1):1–48, 1983.

6 Kazuyuki Amano. Bounds on the Size of Small Depth Circuits for Approximating Majority. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 59–70, 2009.

7 Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the Time-Complexity of Broadcast in Multi-hop Radio Networks: An Exponential Gap Between Determinism and Randomization. *J. Comput. Syst. Sci.*, 45(1):104–126, 1992.

8 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to Compress Interactive Communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013.

9 Eric Blais, Li-Yang Tan, and Andrew Wan. An inequality for the Fourier spectrum of parity decision trees. *CoRR*, abs/1506.01055, 2015. arXiv:1506.01055.

10 Philipp Brandes, Marcin Kardas, Marek Klonowski, Dominik Pajkak, and Roger Wattenhofer. Approximating the Size of a Radio Network in Beeping Model. In Jukka Suomela, editor, *Structural Information and Communication Complexity*, pages 358–373, 2016.

11 Joshua Brody and Elad Verbin. The Coin Problem and Pseudorandomness for Branching Programs. In *Symposium on Foundations of Computer Science (FOCS)*, pages 30–39, 2010.

12 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

13 Ioannis Caragiannis, Clemente Galdi, and Christos Kaklamanis. Basic Computations in Wireless Networks. In *Algorithms and Computation*, 2005.

14 Keren Censor-Hillel, Bernhard Haeupler, Nancy A. Lynch, and Muriel Médard. Bounded-Contention Coding for the additive network model. *Distributed Computing*, 28(5):297–308, 2015.

15 Binbin Chen, Ziling Zhou, and Haifeng Yu. Understanding RFID Counting Protocols. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 291–302, 2013.

16 Gil Cohen, Anat Ganor, and Ran Raz. Two Sides of the Coin Problem. In *Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM)*, pages 618–629, 2014.

17 Alejandro Cornejo and Fabian Kuhn. Deploying Wireless Networks with Beeps. In *International Conference on Distributed Computing (DISC)*, pages 148–162, 2010.

18 Fabien Dufoulon, Janna Burman, and Joffroy Beauquier. Beeping a Deterministic Time-Optimal Leader Election. In *International Symposium on Distributed Computing (DISC)*, pages 20:1–20:17, 2018.

19 Klaus-Tycho Förster, Jochen Seidel, and Roger Wattenhofer. Deterministic Leader Election in Multi-hop Beeping Networks. In *International Symposium on Distributed Computing (DISC)*, pages 212–226, 2014.

20 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

21 Seth Gilbert and Calvin Newport. The Computational Power of Beeps. In *Distributed Computing*, pages 31–46, 2015.

22 John Hastad. Almost Optimal Lower Bounds for Small Depth Circuits. *Advances in Computing Research*, 5:143–170, 1989.

23 Stephan Holzer and Nancy A. Lynch. Brief announcement: Beeping a maximal independent set fast, 2017.

24 Bojun Huang and Thomas Moscibroda. Conflict Resolution and Membership Problem in Beeping Channels. In *International Symposium on Distributed Computing (DISC)*, pages 314–328, 2013.

25 Tomasz Jurdziński, Mirosław Kutyłowski, and Jan Zatopiański. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *Computing and Combinatorics*, pages 279–289, 2002.

26 Jedrzej Kabarowski, Mirosław Kutyłowski, and Wojciech Rutkowski. Adversary Immune Size Approximation of Single-Hop Radio Networks. In *Theory and Applications of Models of Computation*, 2006.

27 Marek Klonowski and Kamil Wolny. Immune Size Approximation Algorithms in Ad Hoc Radio Network. In *Wireless Sensor Networks*, 2012.

28 Gillat Kol, Rotem Oshman, and Dafna Sadeh. Interactive Compression for Multi-Party Protocol. In *International Symposium on Distributed Computing (DISC)*, pages 31:1–31:15, 2017.

29 Zhiyu Liu and Maurice Herlihy. Approximate Local Sums and Their Applications in Radio Networks. In *Distributed Computing*, pages 243–257, 2014.

30 Yves Métivier, John Michael Robson, and Akka Zemmari. On Distributed Computing with Beeps. *CoRR*, abs/1507.02721, 2015. `arXiv:1507.02721`.

31 Saket Navlakha and Ziv Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2015.

32 Calvin Newport and Chaodong Zheng. Approximate Neighbor Counting in Radio Networks. In *International Conference on Principles of Distributed Systems (OPODIS)*, pages 26:1–26:16, 2018.

33 Calvin C. Newport. Radio Network Lower Bounds Made Easy. In *International Conference on Distributed Computing (DISC)*, pages 258–272, 2014.

34 David Peleg. Time-Efficient Broadcasting in Radio Networks: A Review. In *Distributed Computing and Internet Technology (ICDCIT)*, pages 1–18, 2007.

35 Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987.

36 Alex Scott, Peter Jeavons, and Lei Xu. Feedback from nature: an optimal distributed algorithm for maximal independent set selection. In *Symposium on Principles of Distributed Computing (PODC)*, pages 147–156, 2013.

37 Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. In *Symposium on Theory of Computing (STOC)*, pages 589–598, 2008.

38 Roman Smolensky. On Representations by Low-Degree Polynomials. In *Symposium on Foundations of Computer Science (FOCS)*, pages 130–138, 1993.

39 John P. Steinberger. The Distinguishability of Product Distributions by Read-Once Branching Programs. In *Conference on Computational Complexity (CCC)*, pages 248–254, 2013.

40 Andrew Chi-Chih Yao. Separating the Polynomial-Time Hierarchy by Oracles. In *Symposium on Foundations of Computer Science (STOC)*, pages 1–10, 1985.

41 Zhiqiang Zhang and Yaoyun Shi. On the parity complexity measures of Boolean functions. *Theor. Comput. Sci.*, 411(26-28):2612–2618, 2010.