AVERAGE CASE LOWER BOUNDS FOR MONOTONE SWITCHING NETWORKS

by

Robert Robere

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Average Case Lower Bounds for Monotone Switching Networks

Robert Robere

Master of Science

Graduate Department of Computer Science

University of Toronto

2013

An approximate computation of a function $f\colon \{0,1\}^n \to \{0,1\}$ by a computaional model M is a computation in which M computes f correctly on the majority of the inputs (rather than on all inputs). Lower bounds for approximate computations are also known as average case hardness results. We obtain the first average case monotone depth lower bounds for a function in monotone P, tolerating errors that are asymptotically the best possible for monotone circuits. Specifically, we prove average case exponential lower bounds on the size of monotone switching networks for the GEN function. As a corollary, we establish that for every i, there are functions computed with no error in monotone NC^{i+1}, but that cannot be computed without large error by monotone circuits in NC^i.

The work in this paper is joint with Yuval Filmus, Toniann Pitassi and Stephen Cook.

Contents

Chapter 1

Introduction

An approximate computation of a function $f\colon \{0,1\}^n \to \{0,1\}$ by a circuit or switching network M is a computation in which M computes f correctly on the majority of the inputs (rather than on all inputs). More generally, an approximate computation of f with respect to a distribution $\mathcal{D}$ is a computation that computes f correctly on all but an $\epsilon < 1/2$ fraction of the inputs with respect to $\mathcal{D}$. Lower bounds for approximate computation are also known as average case hardness, or correlation bounds. Besides being interesting in their own right, lower bounds for approximate computation have proved useful in many subareas of complexity theory such as cryptography and derandomization. For an excellent survey, see [5].

In this paper, we study the average case hardness of monotone circuits and monotone switching networks (we note here that our results appear in [11]). The first superpolynomial lower bounds on monotone circuit size is the celebrated result due to Razborov [24], who showed that the clique function requires exponential-size monotone circuits, and thus also requires large monotone depth. His result was improved and generalized by many authors to obtain other exponential lower bounds for monotone circuits (for example [1, 2, 4, 14, 15]). All of these bounds are average case lower bounds for functions that lie outside of monotone P. The best known average case lower bound for an explicit function is for Andreev's polynomial problem, which is $(1/2 - 1/n^{1/6})$-hard for subexponential-size circuits (follows from [4]); that means that there is a subexponential function $f(n)$ such that every circuit of size at most $f(n)$ differs from Andreev's polynomial problem on at least a $(1/2 - 1/n^{1/6})$-fraction of the inputs.

Beginning in 1990, lower bound research was aimed at proving monotone size/depth tradeoffs for *efficient* functions that lie inside monotone P. The first such result, due to Karchmer and Wigder-

son [18], established a beautiful equivalence between (monotone) circuit depth and the (monotone) communication complexity of a related communication game. They used this framework to prove that the NL-complete directed connectivity problem requires $\Omega(\log^2 n)$ monotone circuit depth, thus proving that monotone NL (and thus also monotone NC^2) is not contained in monotone NC^1. Subsequently Grigni and Sipser [13] used the communication complexity framework to separate monotone logarithmic depth from monotone logarithmic space. Raz and McKenzie [22] generalized and improved these lower bounds by defining an important problem called the GEN problem, and proved tight lower bounds on the monotone circuit depth of this problem. As corollaries, they separated monotone NC^i from monotone NC^{i+1} for all i, and also proved that monotone NC is a strict subset of monotone P. Unlike the earlier results (for functions lying outside of P), the communication-complexity-based method developed in these papers seems to work only for exact computations.

Departing from the communication game methodology from the 1990's, Potechin [21] recently introduced a new Fourier-analytic framework for proving lower bounds for monotone switching networks. Potechin was able to prove using his framework a $n^{\Omega(\log n)}$ size lower bound for monotone switching networks for the directed connectivity problem. (A lower bound of $2^{\Omega(t)}$ on the size of monotone switching networks implies a lower bound of $\Omega(t)$ on the depth of monotone circuits, and thus the result on monotone switching networks is stronger.) Recently, Chan and Potechin [8] improved on [21] by establishing a $n^{\Omega(h)}$ size lower bound for monotone switching networks for the GEN function, and also for the clique function. Thus, they generalized most of the lower bounds due to Raz and McKenzie to monotone switching networks. However, again their lower bounds apply only for monotone switching networks that compute the function correctly on every input.

In this paper we obtain the first *average case* lower bounds on the size of monotone switching networks (and thus also the first such lower bounds on the depth of monotone circuits) for functions *inside* monotone P. We prove our lower bounds by generalizing the Fourier-analytic technique due to Chan and Potechin. In the process we first give a new presentation of the original method, which is simplified and more intuitive. Then we show how to generalize the method in the presence of errors, which involves handling several nontrivial obstacles.

We show that GEN is $(1/2 - 1/n^{1/3-\epsilon})$-hard for subexponential-size circuits (under a specific distribution), and that directed connectivity is $(1/2 - 1/n^{1/2-\epsilon})$-hard for $n^{O(\log n)}$-size circuits (also under a specific distribution). The latter result is almost optimal since no monotone function is $(1/2 -$

$\log n/\sqrt{n})$-hard even for $O(n \log n)$-size circuits [20], though this result is under the uniform distribution. A related result shows that for every ϵ there is a function that is $(1/2 - 1/n^{1/2-\epsilon})$-hard for subexponential-size circuits [17]. However, the function in [17] results from amplifying a non-explicit random function, while our functions are explicit and computable in polynomial time.

As a corollary to the above theorem, we separate the levels of the NC hierarchy, as well as monotone C from monotone P, in the average case setting. That is, we prove that for all i, there are monotone functions that can be computed exactly in monotone NC^{i+1} but such that any NC^i circuit computing the same function must have large error. And similarly, there are functions in monotone P but such that any monotone NC circuit must have large error.

This leaves open the question of whether or not the original communication-complexity-based approach due to Karchmer and Wigderson can also be generalized to handle errors. This is a very interesting question, since if this is the case, then average case monotone depth lower bounds would translate to probabilistic communication complexity lower bounds for *search problems*. Developing communication complexity lower bound techniques for search problems is an important open problem because such lower bounds have applications in proof complexity, and imply integrality gaps for matrix-cut algorithms (See [3, 16].) We show as a corollary of our main lower bound that the communication complexity approach does not generalize to the case of circuits that make mistakes.

The outline for the rest of the paper is as follows. In Chapter 2 we give background information on switching networks, the GEN function, and Fourier analysis, as well as state our main results (Section 2.5). Then, before proving the lower bound itself, in Section 2.6 we discuss implications for randomized counterparts of the Karchmer-Wigderson construction. In Chapter 3 we give an exponential lower bound for monotone, deterministic switching networks, and Chapter 4 extends it to average case lower bounds. Two appendices describe a reduction from monotone circuits to monotone switching networks (Appendix A) and more intuition for our result on randomized Karchmer-Wigderson games (Appendix B).

Chapter 2

Preliminaries and Main Results

In this chapter we give definitions that will be useful throughout the entire course of the paper. If n is a positive integer then we define the set $[n] = \{1, 2, \ldots, n\}$, and for integers i, j with $i < j$ we define the set $[i, j] = \{i, i+1, \ldots, j\}$ If S is a subset of some universe U, we define $\overline{S} = U \setminus S$ to be the complement of S. As usual, we use $\cup$ to denote the union, $\cap$ to denote the intersection, and $\triangle$ to denote the symmetric difference of two sets. If N and m are positive integers, we denote by $\binom{[N]}{m}$ the collection of all subsets of $[N]$ of size m. The notation $\mathbf{1}$ denotes the vector all of whose entries are 1 (the length of the vector will always be clear from the context). We are interested in studying the space complexity of Boolean functions $f : \{0,1\}^n \to \{0,1\}$. For an input $x \in \{0,1\}^n$, we denote the ith index of x by x_i. For a pair of inputs $x, y \in \{0,1\}^n$, we write $x \leq y$ if $x_i \leq y_i$ for all i. A boolean function f is *monotone* if $f(x) \leq f(y)$ whenever $x \leq y$.

Assume $f\colon \{0,1\}^n \to \{0,1\}$ is a monotone boolean function. An input $x \in \{0,1\}^n$ is called a *maxterm* of f if $f(x) = 0$ and $f(x') = 1$ for the input x' obtained by flipping any 0 in x to a 1. Similarly, an input $x \in \{0,1\}^n$ is called a *minterm* if $f(x) = 1$ and $f(x') = 0$ for the input x' obtained by flipping any 1 in x to a 0. Note that maxterms and minterms are named because they are the **max**imally unsatisfiable and **min**imally satisfiable instances of the boolean function. If f is a boolean function and $f(x) = 1$ we will call x an *accepting* instance of f, otherwise it is a *rejecting* instance.

If $P(x)$ is a boolean condition depending on an input x, then we write $[P(x)]$ to denote the boolean function associated with that condition. For example, if V is a fixed set, we denote by $[U \subseteq V]$ the function $P(U)$ which is 1 if $U \subseteq V$ and 0 otherwise.

2.1 The GEN problem

We will prove lower bounds for the GEN function, originally defined by Raz and McKenzie [22].

Definition 2.1. Let $N \in \mathbb{N}$, and let $L \subseteq [N]^3$ be a collection of triples on $[N]$ called *literals*. For a subset $S \subseteq [N]$, the set of points *generated* from S by L is defined recursively as follows: every point in S is generated from S, and if i, j are generated from S and $(i, j, k) \in L$, then k is also generated from S. (If L were a collection of pairs instead of a collection of triples, then we could interpret L as a directed graph, and then the set of points generated from S is simply the set of points reachable from S.)

The GEN problem is as follows: given a collection of literals L and two distinguished points $s, t \in [N]$, is t generated from $\{s\}$?

For definiteness, in the reminder of the paper we fix (arbitrarily) $s = 1$ and $t = N$.

We assume that every instance of GEN throughout the rest of the paper is defined on the set $[N]$, and we use s and t to denote the start and target points of the instance. Sometimes we want to distinguish a particular literal in instances of GEN, so, if ℓ is a literal appearing in an instance I then we call (I, ℓ) a *pointed instance*.

We can naturally associate GEN instances with some graphs. If G is a DAG with a unique source s, a unique sink t, and in-degree at most 2, then we can form an instance of GEN from G by identifying the start and target points appropriately, and adding a literal (x, y, z) to the GEN instance if the edges $(x, z), (y, z)$ are in G. If z is a vertex of in-degree 1, say $(x, z) \in G$, then we add the literal (x, x, z) instead.

If G does not have a unique source or a unique sink then we can simply add one and connect it to all of the sources or sinks, which is illustrated in Figure 2.3.

The problem GEN is monotone: if we have an instance of GEN given by a set of literals L, and L is an accepting input for GEN, then adding any literal $l \notin L$ to L will not make L a rejecting input. Moreover, it can be computed in monotone P (we leave the proof as an easy exercise).

Theorem 2.2. GEN *is in monotone* P.

Let $(C, \overline{C})$ be a partition of $[N] \setminus \{s, t\}$ into two sets. We call such a set C a *cut* in the point set $[N]$. We think of the cut C as always containing s and never containing t, and so we define $C_s = C \cup \{s\}$.

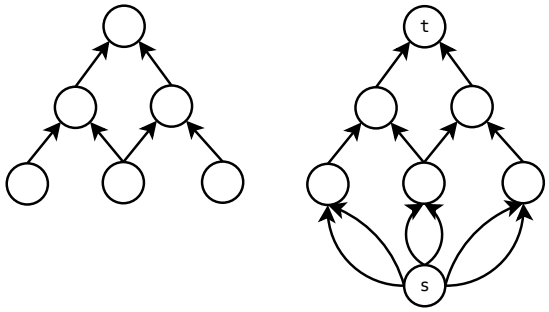

Figure 2.1: A pyramid graph and the corresponding GEN instance

Then we can define an instance $G(C)$ of GEN, called a *cut instance*, as

$$G(C) = [N]^3 \setminus \{(x, y, z) \in N^3 \mid x, y \in C_s, z \in \overline{C}_s\}.$$

If $G(C)$ is a cut instance and $\ell = (x, y, z)$ is a literal with $x, y \in C_s$ and $z \in \overline{C}_s$ then we say that ℓ *crosses* C.

It might seem more natural to define C as a subset of $[N]$ containing s and not containing t. However, from the point of view of Fourier analysis, it is more convenient to remove both "constant" vertices s, t from the equations.

The set of cut instances is exactly the set of maxterms of GEN.

Proposition 2.3. *Let L be an instance of* GEN. *Then L is rejecting if and only if there exists a cut $C \subseteq [N]$ such that $L \subseteq G(C)$.*

Proof. Assume that L is accepting. Then if there existed a cut $C \subseteq [N]$ with $L \subseteq G(C)$, it follows that L would be rejecting from the monotonicity of GEN and the fact that $G(C)$ is a maxterm, which is a contradiction. Similarly, if L is rejecting, then there exists a maxterm $G(C)$ such that $L \subseteq G(C)$, and so for this cut instance there cannot exist a literal $\ell \in L$ with $\ell \notin G(C)$. □

Let $\mathcal{C}$ be the collection of subsets of $[N] \setminus \{s, t\}$, and note that every set $C \in \mathcal{C}$ can be identified with a cut (and therefore a cut instance). We also note that $|\mathcal{C}| = 2^{N-2}$.

2.2 Vectors and Vector Spaces

In this section we recall some definitions from linear algebra. Consider the set of all cuts $\mathcal{C}$ on the point set $[N]$ of GEN. A *cut vector* is a function $f : \mathcal{C} \to \mathbb{R}$, which we think of as a real-valued vector indexed by cuts $C \in \mathcal{C}$. We define an inner product on the space of cut vectors by

$$\langle f, g \rangle = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} f(C)g(C)$$

for any two cut vectors f, g. Two cut vectors f, g are *orthogonal* if $\langle f, g \rangle = 0$, and a set of vectors $\mathcal{V}$ is orthogonal if every pair of vectors in $\mathcal{V}$ is orthogonal. Using this inner product, we define the *magnitude* of a cut vector f to be $||f|| = \sqrt{\langle f, f \rangle}$.

We will also need some tools from Fourier analysis. Given a cut $U \in \mathcal{C}$, define the cut vector $\chi_U : \mathcal{C} \to \mathbb{R}$ by

$$\chi_U(C) = (-1)^{|U \cap C|}.$$

We have the following proposition regarding these vectors:

Proposition 2.4. *The collection of vectors $\{\chi_U\}_{U \in \mathcal{C}}$ is orthonormal.*

Proof. If U and V are two sets with $U = V$, then

$$\langle \chi_U, \chi_V \rangle = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \chi_U(C)\chi_V(C) = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \chi_U(C)^2 = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} (-1)^{2|U \cap C|} = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} 1^{|U \cap C|} = \frac{\mathcal{C}}{\mathcal{C}} = 1.$$

So, let U and V be two vectors with $U \neq V$. Then

$$\begin{aligned} \langle \chi_U, \chi_V \rangle &= \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \chi_U(C)\chi_V(C) = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} (-1)^{|U \cap C|}(-1)^{|V \cap C|} \\ &= \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} (-1)^{|U \cap C| + |V \cap C|} = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} (-1)^{|(U \triangle V) \cap C|}, \end{aligned}$$

where the last equality follows since each element in $U \cap V \cap C$ is counted twice. Since $U \triangle V \neq \emptyset$ by assumption, it follows by cancellation that

$$\frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} (-1)^{|(U \triangle V) \cap C|} = 0$$

and the proof is complete. □

This particular collection of vectors forms a basis for the vector space of cut vectors known as the *Fourier basis*. It follows that we can write any cut vector $f\colon \mathcal{C} \to \mathbb{R}$ as $f = \sum_{C\in\mathcal{C}} \langle f, \chi_C\rangle \chi_C$, where $\langle f, \chi_C\rangle$ is called the *Fourier coefficient* at C. Following convention, we will denote $\langle f, \chi_C\rangle$ by $\hat{f}(C)$.

We need some useful properties of the Fourier transform. First recall *Parseval's Theorem*: let f be any cut vector, then

$$\langle f, f\rangle = \sum_{C\in\mathcal{C}} \hat{f}(C)^2, \tag{2.1}$$

Parseval's theorem also holds in a more general setting: If B is an orthonormal set of cut vectors then

$$\langle f, f\rangle \geq \sum_{\phi\in B} |\langle f, \phi\rangle|^2. \tag{2.2}$$

2.3 Switching Networks

In this section we introduce switching networks, which are a computation model used to capture space-bounded computation.

Definition 2.5. Let $X = \{x_1, \ldots, x_n\}$ be a set of input variables. A *monotone switching network* $\mathbf{M}$ on the variables X is specified as follows. There is an underlying graph undirected $G = (V, E)$ whose nodes are called *states* and whose edges are called *wires*, with a distinguished *start state* $\mathbf{s}$ and a distinguished *target state* $\mathbf{t}$. The wires of $\mathbf{M}$ are labelled with variables from X.

Given an input $x\colon X \to \{0, 1\}$ (or an assignment of the variables), the switching network responds as follows. Let $\mathbf{e}$ be a wire in the switching network, and let x_i be the variable labelling $\mathbf{e}$. The edge $\mathbf{e}$ is *alive* if $x_i = 1$, and it is *dead* if $x_i = 0$.

We say that $\mathbf{M}$ *accepts* an input x if there exists a path from $\mathbf{s}$ to $\mathbf{t}$ using only wires which are alive under x. If no such path exists, then $\mathbf{M}$ *rejects* x. The Boolean function computed by $\mathbf{M}$ is $f(x) = [\mathbf{M} \text{ accepts } x]$.

Throughout the paper we follow the convention used in the previous definition and use **bold** face to denote objects in switching networks.

We briefly mention some computation models extending monotone switching networks. If the underlying graph G is directed, then instead of a switching network we have a switching-and-rectifier

network. A non-monotone switching network can have negated literals on the wires. All switching networks considered in this paper are monotone and undirected.

Monotone switching networks and monotone circuits are connected by the following result essentially proved by Borodin [6]: a monotone circuit of depth d can be simulated by a monotone switching network of size 2^d, and a monotone switching network of size s can be simulated by a monotone circuit of depth $O(\log^2 s)$. (The result holds also for non-monotone switching networks and non-monotone circuits.) See Appendix A for a proof sketch.

A *switching network for* GEN is a switching network whose input is an instance of GEN. Such a switching network is *complete* if it accepts all yes instances of GEN. It is *sound* if it rejects all no instances of GEN. A switching network which is both complete and sound computes the GEN function.

Let $\mathbf{M}$ be a switching network for GEN. We can naturally identify each state $\mathbf{u}$ in the switching network $\mathbf{M}$ with a *reachability vector* $R_{\mathbf{u}} \colon \mathcal{C} \to \{0,1\}$ defined by

$$R_{\mathbf{u}}(C) := \begin{cases} 1 & \text{if } \mathbf{u} \text{ is reachable on input } G(C), \\ 0 & \text{otherwise.} \end{cases}$$

Here are some basic properties of the reachability vectors.

Theorem 2.6. *Let* $\mathbf{M}$ *be a switching network with start state* $\mathbf{s}$ *and target state* $\mathbf{t}$.

1. $R_{\mathbf{s}} \equiv 1$.
2. *If* $\mathbf{M}$ *is sound then* $R_{\mathbf{t}} \equiv 0$.
3. *If* $\mathbf{u}$ *and* $\mathbf{v}$ *are two states connected by a wire labelled* ℓ *and* C *is a cut with* $\ell \in G(C)$ *then* $R_{\mathbf{u}}(C) = R_{\mathbf{v}}(C)$.

Proof. By the above definition, if $\mathbf{s}$ is the start state of the network $\mathbf{M}$ then we have $R_{\mathbf{s}}(C) = 1$ for every cut C since the state $\mathbf{s}$ is reachable on every instance. Similarly, it should be clear from the definition of the reachability vectors that the switching network $\mathbf{M}$ is sound only if $R_{\mathbf{t}}(C) = 0$ for each cut C since the collection of instances $\{G(C) \mid C \in \mathcal{C}\}$ are exactly the maxterms of GEN by Proposition 2.3.

Finally, we show that if $\mathbf{u}$ and $\mathbf{v}$ are two states in $\mathbf{M}$ connected by a wire labelled with ℓ, and if C is a cut with $\ell \in G(C)$, then $R_{\mathbf{u}}(C) = R_{\mathbf{v}}(C)$. For observe that if $\ell \in G(C)$ then the wire labelled with

ℓ connecting $\mathbf{u}$ and $\mathbf{v}$ will be alive during the computation of $\mathbf{M}$ on $G(C)$, and so the state $\mathbf{u}$ will be reachable if and only if the state $\mathbf{v}$ is. □

2.4 Reversible Pebbling for GEN

Next we discuss the reversible pebbling game on graphs, which is a space-efficient way to perform reachability tests on graphs. The particular form of this test gives an algorithm for GEN by applying it to the underlying graph of a GEN instance.

Definition 2.7. Let $G = (V, E)$ be a directed acyclic graph (DAG) with a unique source s and a unique sink t. For a node $v \in V$, let $P(v) = \{u \in V : (u, v) \in E\}$ be the set of all incoming neighbors of v. We define the *reversible pebbling game* as follows. A *pebble configuration* is a subset $S \subseteq V$ of "pebbled" vertices. For every $x \in V$ such that $P(x) \subseteq S$, a *legal pebbling move* consists of either pebbling or unpebbling x, see Figure 2.2. Since s is a source, $P(s) = \emptyset$, and so we can always pebble or unpebble it.

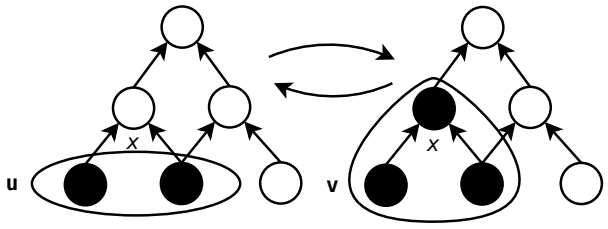

Figure 2.2: Legal pebbling moves involving x; the corresponding pebbling configurations are u and v

The goal of the reversible pebbling game is to place a pebble on t, using only legal pebbling moves, starting with the empty configuration, while minimizing the total number of pebbles used simultaneously. Formally, we want to find a sequence of pebbling configurations $S_0 = \emptyset, S_1, \ldots, S_n$ such that $t \in S_n$, and for each $i \in \{0, \ldots, n-1\}$, the configuration S_{i+1} is reachable from configuration S_i by a legal pebbling move. We call such a sequence a *valid pebbling sequence* for G. The *pebbling cost* of the sequence is $\max(|S_0|, \ldots, |S_n|)$. The *reversible pebbling number* of a DAG G is the minimal pebbling cost of a valid pebbling sequence for G.

Dymond and Tompa [10] showed that the reversible pebbling number of *any* graph with n vertices

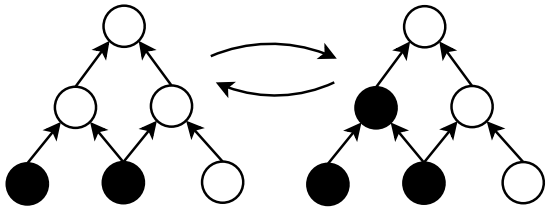

Figure 2.3: Reversible Pebbling Moves on the Pyramid

and in-degree 2 is $O(n/\log n)$. Gilbert and Tarjan [12] constructed matching graphs (see also Nordström's excellent survey [19]).

Theorem 2.8. *There is an explicit family of DAGs G_n with in-degree 2 such that G_n has n vertices and reversible pebbling number $\Omega(n/\log n)$.*

In fact, the graphs G_n have *black-white* pebbling number $\Omega(n/\log n)$. For more on reversible pebbling and its applications, see [7].

Pyramid graphs are graphs with $O(h^2)$ nodes and reversible pebbling number $\Theta(h)$ for each positive integer h.

Definition 2.9. A directed graph $P = (V, E)$ is a *pyramid graph with h levels* if V is partitioned into h subsets, $V_1, V_2, \ldots, V_h$ (called levels), where V_i has i vertices. Let $V_i = \{v_{i1}, v_{i2}, \ldots, v_{ii}\}$. For each $i \in [h-1]$, if v_{ij} and $v_{i,j+1}$ are a pair of adjacent vertices in layer V_i, then there are edges $(v_{ij}, v_{i+1,j-i-1})$ and $(v_{i,j+1}, v_{i+1,j-i-1})$.

Cook [9] calculated the pebbling number of pyramids.

Theorem 2.10. *Let P be any pyramid graph with h levels. The reversible pebbling number of P is $\Theta(h)$.*

Another useful class of graphs are path graphs.

Definition 2.11. A directed graph $P = (V, E)$ is a *directed path of length n* if $V = \{v_1, \ldots, v_n\}$ and $E = \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}$.

Potechin [21] computed the reversible pebbling number of directed paths. In hindsight, the same computation appears in Raz and McKenzie [22]; they computed a different statistic of the graph, which was shown to equal the reversible pebbling number by Chan [7].

Theorem 2.12. *The reversible pebbling number of a directed path of length n is $\Theta(\log n)$.*

Every DAG with in-degree at most 2 naturally defines a minterm of GEN which we call a *graph instance*.

Definition 2.13. Let G be a DAG with in-degree at most 2 and a single sink t, and suppose the vertex set of G is a subset of $[N]$ not containing s. The GEN instance corresponding to G contains the following triples: for each source x, the triple (s, s, x); for each vertex z with inbound neighborhood $\{x\}$, the triple (x, x, z); for each vertex z with inbound neighborhood $\{x, y\}$, the triple (x, y, z). Such an instance is called a *graph instance* isomorphic to G. The *underlying vertex set* consists of the vertex set of G with the vertex t removed.

For a graph G, the function $G - \mathsf{GEN}$ is the monotone function whose minterms are all graph instances isomorphic to G.

Raz and McKenzie [22] proved the following result.

Theorem 2.14. *For each DAG G there is a polynomial size monotone circuit of depth $\Theta(h \log N)$ for $G - \mathsf{GEN}$, where h is the reversible pebbling number of G.*

Their main contribution was to prove a matching lower bound on circuit depth, for $m = N^{O(1)}$.

2.5 Statement of Main Results

In this paper we prove lower bounds for monotone switching networks computing GEN. Our first contribution is a simplified proof of the following theorem [8] which gives an exponential lower bound for deterministic monotone switching networks.

Theorem 2.15. *Let N, m, h be positive integers, and let G be a DAG on $m+1$ vertices with in-degree at most 2 and reversible pebbling number at least $h + 2$. Any sound monotone switching network for* GEN *which accepts all graph instances isomorphic to G must have at least $\Omega(hN/m^2)^{h/3}/O(m)$ states.*

Corollary 2.16. *For any $\epsilon > 0$, any monotone switching network which computes* GEN *must have at least $2^{\Omega(\epsilon N^{1-\epsilon})}$ states.*

Proof. Apply the theorem with $m = N^{1-\epsilon}$ and a DAG G with reversible pebbling number $h = \Theta(m/\log m)$, given by Theorem 2.8. □

We also consider monotone switching networks which are allowed to make errors. Let $\mathcal{D}$ be any distribution on instances of GEN. We say that a monotone switching network $\mathbf{M}$ computes GEN with error ϵ if the function computed by $\mathbf{M}$ differs from GEN on an ϵ-fraction of inputs (with respect to $\mathcal{D}$).

The distributions $\mathcal{D}$ we use are parametrized by DAGs. For any DAG G with in-degree at most 2 we define $\mathcal{D}_G$ to be the distribution on instances of GEN which with probability $1/2$ chooses $G(C)$ for a uniformly random cut $C \in \mathcal{C}$, and with probability $1/2$ chooses a uniformly random graph instance isomorphic to G.

Our major result in this paper is a strong extension of Theorem 2.15 for switching networks computing GEN with error close to $1/2$.

Theorem 2.17. *Let α be a real number in the range $0 < \alpha < 1$. Let m, h, N be positive integers satisfying $324m^2 \leq N^\alpha$, and let G be a DAG with $m+1$ vertices, in-degree 2 and reversible pebbling number at least $h+2$.*

Any monotone switching network which computes GEN on $[N+2]$ with error $\epsilon \leq 1/2 - 1/N^{1-\alpha}$ must have at least $\Omega(hN/m^2)^{h/3}/O(mN)$ states.

Corollary 2.18. *For any α in the range $0 < \alpha < 1$, any monotone switching network computing GEN with error at most $1/2 - 1/N^{1-\alpha}$ must have at least $2^{\Omega(\alpha N^{\alpha/2})}$ states.*

Proof. Apply the theorem with $m = N^{\alpha/2}/324$ and a DAG G with reversible pebbling number $h = \Theta(m/\log m)$, given by Theorem 2.8. □

Using this theorem, we get the following corollary separating NC^i and NC^{i+1} in the presence of errors. Recall from Theorem 2.10 that the pyramid graph with height h has reversible pebbling number $\Theta(h)$ and $m = h(h+1)/2$ nodes.

Theorem 2.19. *Let $0 < \delta < 1/3$ be any real constant. For each positive integer i there exists a language L which is computable in monotone NC^{i+1}, but there is no sequence of circuits in monotone NC^i which computes L on inputs of length k with error $\epsilon \leq 1/2 - 1/k^{1/3-\delta}$.*

Proof. For each N, let $L_{N^3} \subseteq \{0,1\}^{N^3}$ be $G - \mathsf{GEN}$, where G is a pyramid of height $h = \log^i N$, and let $L = \bigcup_{N>0} L_{N^3}$. Theorem 2.14 shows that for each N there exists a polynomial size circuit of depth $O(h \log N) = O(\log^{i+1} N)$ computing L_{N^3}, and so L is in monotone NC^{i+1}.

On the other hand, any monotone circuit C with bounded fan-in and depth d can be simulated by a monotone switching network with 2^d states. Therefore Theorem 2.17 (with $\alpha = 3\delta$) implies that for large enough N, any monotone circuit computing L_{N^3} with $\epsilon \leq 1/2 - 1/N^{1-3\delta} \leq 1/2 - 1/k^{1/3-\delta}$ error must have depth $\Omega(h \log N) = \Omega(\log^{i+1} N)$. □

Similarly, we can separate monotone NC from monotone P.

Theorem 2.20. *Let $0 < \delta < 1/3$ be any real constant. There exists a language L which is computable in monotone* P*, but there is no sequence of circuits in monotone* NC *which computes L on inputs of length k with error $\epsilon \leq 1/2 - 1/k^{1/3-\delta}$.*

Proof. The proof is similar to the proof of the previous theorem. Instead of choosing $h = \log^i N$, choose $h = N^{1/100}$. □

We also get the following result for directed connectivity which approaches the optimal result mentioned in the introduction (albeit with a different input distribution).

Theorem 2.21. *Let $0 < \delta < 1/2$ be any real constant. For $k = N^2$, let f be the function whose input is a directed graph on N vertices, and $f(G) = 1$ if in the graph G, the vertex N is reachable from the vertex 1. There exists a distribution $\mathcal{D}$ on directed graphs such that any monotone switching network computing f with error $\epsilon < 1/2 - 1/k^{1/2-\delta}$ (with respect to $\mathcal{D}$) contains at least $N^{\Omega(\log N)}$ states.*

Proof. We can reduce f to GEN by replacing each edge (u, v) in the input graph by a triple (u, u, v). The resulting instance is accepted by GEN iff $t := N$ is reachable from $s := 1$ in the input graph.

Given N, let G_N be the directed path of length $N^{1/100}$. Theorem 2.12 shows that G_N has reversible pebbling number $\Theta(\log N)$. Applying Theorem 2.17 (with $\alpha = 2\delta$) yields a lower bound of $N^{\Theta(\log N)}$ on the size of monotone switching networks computing f with error $\epsilon < 1/2 - 1/N^{2\delta} = 1/2 - 1/k^{\delta}$ with respect to the distribution $\mathcal{D}_{G_N}$. □

2.6 On Randomized Karchmer-Wigderson

For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ let $R_f \subseteq \{0,1\}^{2n} \times [n]$ be the following relation associated with f. If $\alpha \in f^{-1}(1)$ and $\beta \in f^{-1}(0)$, then $R_f(\alpha, \beta, i)$ holds if and only if $\alpha(x_i) \neq \beta(x_i)$. Intuitively,

if α is a 1 of the function and β is a 0 of the function, then i is an index where α and β differ. Similarly, if f is a monotone Boolean function, then $R_f^m(\alpha, \beta, i)$ if and only if $\alpha(x_i) = 1$ and $\beta(x_i) = 0$.

Karchmer and Wigderson [18] proved that for any Boolean function, the minimum depth of any circuit computing f is equivalent to the communication complexity of R_f, and similarly for any monotone Boolean function, the minimum monotone circuit depth for f is equivalent to the communication complexity of R_f^m.

We are interested in whether there is an analog of the Karchmer-Wigderson result in the case of circuits and communication complexity protocols that make errors. That is, is it true that for any Boolean function f, the minimum depth of any circuit that computes f correctly on most inputs over a distribution $\mathcal{D}$, is related to the communication complexity for R_f with respect to $\mathcal{D}$? And similarly is monotone circuit depth related to communication complexity in the average case setting?

Such a connection would be very nice for several reasons. First, a communication complexity approach to proving average case circuit lower bounds is appealing. Secondly, proving lower bounds on the average case communication complexity of relations is an important problem as it is related to proving lower bounds in proof complexity. In particular, average case communication complexity lower bounds for relations associated with unsatisfiable formulas imply lower bounds for several proof systems, including the cutting planes and Lová sz-Schrijver systems [3, 16]. Thus an analog of Karchmer-Wigderson in the average case setting, together with our new lower bounds for monotone circuits, would imply a new technique for obtaining average case communication complexity lower bounds for search problems.

In the forward direction, it is not hard to see that if C is a circuit that computes f correctly on all but an ϵ fraction of inputs, then there is a communication complexity protocol for R_f (or for R_f^m in the case where C is a monotone circuit) with low error. If the circuit C on inputs α and β yields the correct answer, then the Karchmer-Wigderson protocol will be correct on the pair α, β.

However, the reverse direction is not so clear. Raz and Wigderson [23] showed that in the non-monotone case, the Karchmer-Wigderson connection is false. To see this, we note that given inputs α, β, there is an $O(\log^2 n)$ randomized protocol that finds a bit i such that $\alpha_i \neq \beta_i$, or that determines that $\alpha = \beta$. (Recursively apply the randomized equality testing protocol on the left/right halves of the inputs in order to find a bit where the strings are different, if such an index exists; a more efficient protocol along similar lines has complexity $O(\log n)$.) Using this protocol, it follows that for every Boolean

function, R_f has an efficient randomized protocol. Now by Yao's theorem, this implies that for every function f and every distribution $\mathcal{D}$, there is an efficient protocol for R_f with low error with respect to $\mathcal{D}$. On the other hand, by a counting argument, almost all Boolean functions require exponential-size circuits even to approximate.

The intuitive reason that the Karchmer-Wigderson reduction works only in the direction circuits to protocols is that protocols are cooperative while circuits are, in some sense, competitive. We make this argument more precise in Appendix B.

In the monotone case, it is not as easy to rule out an equivalence between randomized monotone circuit depth for f and randomized communication complexity for R_f^m. The above argument for the non-monotone case breaks down here because the communication problem of finding an input i such that $\alpha_i > \beta_i$, given the promise that such an i exists, is equivalent to the promise set disjointness problem where Alice and Bob are given two sets with the promise that they are not disjoint, and they should output an element that is in both sets. Promise set disjointness is as hard as set disjointness by the following reduction. Given an efficient protocol for promise set disjointness, and given an instance α, β of R_f^m where $|\alpha| = |\beta| = n$, the players create strings α', β' of length $n+1$ by selecting an index $i \in [n+1]$ randomly, and inserting a 1 in position i into both α and β. If α and β were disjoint, running a protocol for promise disjointness on α' and β' is guaranteed to return the index of the planted 1. Otherwise, the promise disjointness protocol should return a different index (not the planted one) with probability at least $1/2$. Repeating logarithmically many times yields a protocol that solves set disjointness with low error.

We will now prove that the Karchmer-Wigderson equivalence is also false in the monotone case.

Theorem 2.22. *The monotone Karchmer-Wigderson reduction does not hold in the average-case setting. That is, there is a monotone function $f\colon \{0,1\}^n \to \{0,1\}$, a distribution $\mathcal{D}$ and an ϵ satisfying $0 < \epsilon < 1/2$, such that there is an efficient protocol for R_f^m with error at most ϵ with respect to $\mathcal{D}$ but such that any subexponential-size monotone circuit for f of depth n^ϵ has error greater than ϵ with respect to $\mathcal{D}$.*

Proof. We will consider the GEN problem on a universe of size N. Our distribution will consist of a distribution of minterms and maxterms. The minterms will be pyramids of size n, and the maxterms will be all cuts. We give an efficient randomized protocol for solving R_f^m on the uniform distribution over minterms and maxterms. A random pyramid P is generated by choosing m points uniformly at

random from $[N]$ and constructing the pyramid triples arbitrarily. (For example, pick a permutation of the m points, and let this define the pyramid triple.) Add the triples (s, s, v_i) for all $1 \leq i \leq h$, where v_i, $i \in [h]$ are the elements at the bottom of the pyramid, and add the triple (u, u, t), where u is the element at the top of the pyramid. A random cut C is generated by choosing each $i \in [N]$ to be on the s-side with probability $1/2$, and otherwise on the t-side. The instance corresponding to C is obtained by adding all triples except for those triples that "cross the cut" – that is, the triples (i, j, k) where $i, j \in C$ and $k \in \overline{C}$.

Now consider the following protocol for solving R_f^m. On input P for Alice, Alice deterministically selects a set of $\log m$ disjoint triples (i, j, k) in P and sends them to Bob. Then Bob checks whether or not any of these triples crosses his cut, and if so he outputs one of these triples, and otherwise the protocol fails, and Bob outputs an arbitrary triple.

We will call a cut C *bad* for P if the above algorithm fails. For each P, we will upper bound the fraction of cuts that are bad for P. If (i, j, k) is a triple appearing in P, then the probability over all cuts C that (i, j, k) does not cross C is at most $1/8$ (since it is the probability that i, j lie on the s-side, and k lies on the t-side). Since we have $\log m$ disjoint triples, the probability that none of them cross C is at most $1/m$. Overall, our protocol has communication complexity $O(\log^2 m)$ and solves R_f^m over our distribution of instances with probability at least $1 - 1/m$.

On the other hand, by our main lower bound (cf. Corollary 2.16), any small monotone circuit for solving GEN errs with superconstant probability over this same distribution of instances.

□

We note that the hard examples that were previously known to be hard for monotone circuits with errors, such as clique/coclique, are not good candidates for the above separation. To see this note that if yes instances are k-cliques and no instances are $k - 1$ colorings, then the clique player needs to send about $\sqrt{k}$ vertices in order to have a good chance of finding a repeated color, and thus the protocol is not efficient in the regime where clique/coclique is exponentially hard (for $k = n^\epsilon$).

We also note that our result implies that the proof technique of [22] does not generalize to prove average case lower bounds for monotone circuits. This leaves the interesting open question of establishing a new connection between randomized circuit depth and communication complexity. One possibility is to consider randomized protocols for R_f or R_f^m which succeed with high probability on *all* pairs of

inputs (the randomized counterpart of the protocol in the proof of Theorem 2.22 fails in some extreme cases), and connect them to some "distribution-less" notion of randomized circuits.

Chapter 3

Simplified Lower Bounds for Exact Computation

In this section we give a simplified proof of the main theorem from [8], which we restate here[1].

Theorem 2.15. *Let N, m, h be positive integers, and let G be a DAG on $m+1$ vertices with in-degree at most 2 and reversible pebbling number at least $h+2$. Any sound monotone switching network for* GEN *which accepts all graph instances isomorphic to G must have at least $\Omega(hN/m^2)^{h/3}/O(m)$ states.*

3.1 Overview of Proof

We focus on a set of minterms and maxterm over a ground set of size N. The minterms are height h, size m *pyramids*[2], where $m = O(N^{1/3})$, and the maxterms are *cuts*, given by a subset C of the vertices containing s and not containing t. The instance $G(C)$ corresponding to C contains all triples except for (i, j, k) where i and j are in C and k is not in C. Our minterms will consist of a special exponential-sized family of pyramid instances, $\mathcal{P}$, with the property that their pairwise intersection is at most h, and our maxterms, $\mathcal{C}$, will consist of all cuts. Given a monotone switching network $(\mathbf{M}, \mathbf{s}, \mathbf{t})$ solving GEN over $[N]$, for each state $\mathbf{v}$ we define its reachability function $R_{\mathbf{v}} : \mathcal{C} \to \{0, 1\}$ given by $R_{\mathbf{v}}(C) = 1$ if v is reachable from s in the instance $G(C)$, and otherwise $R_v(C) = 0$.

[1] The result proved in [8] is stated only for pyramids, but as noted in [7], the proof works for any DAG with in-degree at most 2 once we replace h with the reversible pebbling number of the graph.

[2] Our proof is presented more generally for any fixed graph of size roughly N^{ϵ} but for simplicity of the proof overview, we will restrict attention to pyramid yes instances.

At the highest level, the proof is a bottleneck counting argument. For each pyramid $P \in \mathcal{P}$, we will construct a function g_P from the set $\mathcal{C}$ of all cuts to the reals. This function will satisfy three properties.

(1) For every $P \in \mathcal{P}$ there is a "complex" state v_P in the switching network such that $\langle g_P, R_{\mathbf{v}_P} \rangle = \Omega(1/|\mathbf{M}|)$.

(2) g_P only depends on coordinates from P, and g_P has zero correlation with any function which depends on at most h coordinates.

(3) Finally, $\|g_P\|$ is upper-bounded by $m^{O(h)}$.

The first property tells us that for every pyramid $P \in \mathcal{P}$, there is a complex state $\mathbf{v}_P$ in the network that is specialized for P. The second property, together with the fact that the P's in $\mathcal{P}$ are pairwise disjoint, will imply that the functions $\{g_P : P \in \mathcal{P}\}$ are orthogonal, and thus we have

$$1 = \langle R_{\mathbf{v}}, R_{\mathbf{v}} \rangle \geq \sum_{P \in \mathcal{P}} |\langle R_{\mathbf{v}}, \frac{g_P}{\|g_P\|} \rangle|^2 = \frac{1}{m^{O(h)}} \sum_{P \in \mathcal{P}} \langle g_P, R_{\mathbf{v}} \rangle^2.$$

By the third property, $\|g_P\|$ is small, and thus it follows that no state $\mathbf{v}$ cannot be complex for more than $N^2 m^{O(h)}$ different P's (since otherwise, the quantity on the right side of the above equation would be greater than 1.) This together with the fact that $|\mathcal{P}|$ is very large, so that $|\mathcal{P}|/(N^2 m^{O(h)})$ is still exponentially large, imply our lower bound.

It remains to construct these magical functions g_P. For the rest of this overview, fix a particular pyramid P. How can we show that some state in the switching network is highly specific to P? We will use the original Karchmer-Wigderson intuition which tells us that in order for a monotone circuit to compute GEN correctly (specifically, to output "1" on P and "0" on all cuts), it must for every cut C produce a witness literal $l \in P$ that is not in $G(C)$. And (intuitively) because different literals must be used as witnesses for different cuts, this should imply that a non-trivial amount of information must be remembered in order to output a good witness. This intuition also occurs in many information complexity lower bounds.

The above discussion motivates studying progress (with respect to our fixed pyramid P, and all cuts), by studying progress of the associated search problem. To this end, for each pyramid $P \in \mathcal{P}$ and literal $\ell \in P$, we will consider ℓ-nice functions $g_{P,l}$, where ℓ-nice means that $g_{P,\ell}(C) = 0$ whenever $G(C)(\ell) = 1$. We will think of an ℓ-nice function $g_{P,\ell}$ as a "pseudo" probability distribution over

no instances (cuts) that puts zero mass on all cuts C such that $G(C)(\ell) = 1$. ($g_{P,\ell}$ is not an actual distribution since it attains both positive and negative values.) For a state $\mathbf{u}$ in the switching network, the inner product $\langle R_{\mathbf{u}}, g_{P,\ell} \rangle$ will be our measure of progress of the GEN function with respect to the pyramid P, on no instances where the witness *cannot* be ℓ. In order for $g_{P,\ell}$ to behave like a distribution, we will require that $\langle g_{P,\ell}, 1 \rangle = 1$

Because $R_{\mathbf{s}}$ accepts all cuts, it follows that $\langle R_{\mathbf{s}}, g_{P,\ell} \rangle = 1$, which we interpret as saying that at the start state, we have made no progress on rejecting the pseudo-distribution defined by $g_{P,\ell}$. Similarly, because $R_{\mathbf{t}}$ rejects all cuts, it follows that $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle = 0$, which we interpret as saying that at the final state, we have made full progress since we have successfully rejected the entire pseudo-distribution defined by $g_{P,\ell}$.

For our yes instance P and some literal $\ell \in P$, let $\mathbf{p} = \mathbf{s}, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q, \mathbf{t}$ be an accepting computation path on P. If we trace the corresponding inner products along the path, $\langle R_{\mathbf{s}}, g_{P,\ell} \rangle$, $\langle R_{\mathbf{u}_1}, g_{P,\ell} \rangle$, $\ldots$, $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle$, they will start with value 1 and go down to 0. Now if $\mathbf{u}$ and $\mathbf{v}$ are adjacent states in the switching network connected by the literal ℓ, then progress at $\mathbf{u}$ with respect to $g_{P,\ell}$ is the same as progress at $\mathbf{v}$ with respect to $g_{P,\ell}$. This is because the pseudo-distribution defined by $g_{P,\ell}$ ignores inputs where ℓ crosses the cut (they have zero "probability"), and all other cuts reach $\mathbf{u}$ iff they reach $\mathbf{v}$.

This allows us to invoke a crucial lemma that we call the gap lemma, which states the following. Fix an accepting path $\mathbf{p}$ for the yes instance P. Then since for every $\ell \in P$, $\langle g_{P,\ell}, R_{\mathbf{s}} \rangle = 1$ and $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle = 0$, and for every pair of adjacent states $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ along the path, one of these inner products doesn't change, then there must exist some node $\mathbf{v}$ on the path and two literals $\ell_1, \ell_2 \in P$ such that $|\langle g_{P,\ell_1}, R_{\mathbf{v}} \rangle - \langle g_{P,\ell_2}, R_{\mathbf{v}} \rangle| \geq 1/|M|$. Thus the gap lemma for P implies that this state $\mathbf{v}$ behaves significantly differently on the two pseudo-distributions g_{P,ℓ_1} and g_{P,ℓ_2} of cuts, and therefore this node can distinguish between these two pseudo-distributions. We will let $g_P = g_{P,\ell_1} - g_{P,\ell_2}$ be the pseudo-distribution associated with P. In summary, the gap lemma implies that for every yes instance P, we have a pseudo-distribution g_P and a "complex state" in the switching network which is highly specific to g_P. Thus we have shown property (1) above.

In order to boost the "complex state" argument and get an exponential size lower bound, as explained earlier, we still need to establish properties (2) and (3). The construction proceeds in two steps: first we construct functions $g_{P,\ell}$ satisfying properties (2) but whose norm is too large, and then we fix the norm. Our construction is the same as in earlier papers, and this is the essential place in the proof where

the pebbling number of the graph comes into play. (For pyramid graphs, the pebbling number is $\Theta(h)$, where h is the height of the pyramid.) The construction, while natural, is technical and thus we defer its explanation to Appendix 3.2.

Now we want to generalize the above argument to switching networks that are allowed to make errors. Several important things go wrong in the above argument. First, the set $\mathcal{P}$ of pyramids that we start with above may not be accepted by the network, and in fact it might even be that none of them are accepted by the network. Secondly, it is no longer true that $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle = 0$ as required in order to apply the gap lemma, because now there may be many cuts which are incorrectly accepted by the switching network.

The first problem can be easily fixed since if a random pyramid is accepted by the network, then we can still find a large design consisting of good pyramids, by taking a random permutation of a fixed design. Solving the second problem is more difficult. In the worst case, it may be that $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle \neq 0$ for all $g_{P,\ell}$, and so the gap lemma cannot be applied at all. To address this issue, we will say that a pyramid is *good* for a network if it is both accepted by the network, and if $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle$ is small (say less than $1/2$) for *some* $\ell \in P$. We are able to prove (by estimating the second moment) a good upper bound on the probability that $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle$ is large, and thus we show that with constant probability, a random pyramid is good. Then we generalize our gap lemma to obtain an "approximate" gap lemma which essentially states that as long as the inner products with $R_{\mathbf{t}}$ are not too close to 1, then we can still find a complex state for P in the network; in fact, it is enough that *one* inner product is not too close to 1. Using these two new ingredients, we obtain average case exponential lower bounds for monotone switching networks for GEN.

3.2 Construction of Nice Vectors

As discussed in the overview, the proof makes use of ℓ-nice vectors, which we proceed to define.

Definition 3.1. Let $g \colon \mathcal{C} \to \mathbb{R}$ be a cut vector and $\ell \in [N]^3$. Recall that for each cut C in GEN we associate a cut instance $G(C)$.

We say that g is *ℓ-nice* if $\langle g, \mathbf{1} \rangle = 1$ and $g(C) = 0$ whenever $\ell \notin G(C)$.

In a sense, vectors which are ℓ-nice are "ignorant" of cuts which are crossed by the literal ℓ. This has the following implication.

Lemma 3.2. *Let $\ell \in [N]^3$ and let $\mathbf{M}$ be a monotone switching network for* GEN*. If a cut vector g is ℓ-nice then for any pair of states $\mathbf{u}, \mathbf{v} \in \mathbf{M}$ connected by a wire labelled ℓ we have $\langle R_{\mathbf{u}}, g\rangle = \langle R_{\mathbf{v}}, g\rangle$.*

Proof. Suppose $\mathbf{u}, \mathbf{v} \in \mathbf{M}$ are connected by a wire labelled ℓ. Theorem 2.6 shows that $R_{\mathbf{u}}(C) = R_{\mathbf{v}}(C)$ whenever $\ell \in G(C)$. Since $g(C) = 0$ whenever $\ell \notin G(C)$,

$$\langle R_{\mathbf{u}}, g\rangle = \frac{1}{|\mathcal{C}|} \sum_{\substack{C \in \mathcal{C} \\ \ell \in G(C)}} R_{\mathbf{u}}(C) g(C) = \frac{1}{|\mathcal{C}|} \sum_{\substack{C \in \mathcal{C} \\ \ell \in G(C)}} R_{\mathbf{v}}(C) g(C) = \langle R_{\mathbf{v}}, g\rangle. \qquad \square$$

For each graph instance P of GEN we will come up with ℓ-nice functions $g_{P,\ell}$ for each $\ell \in P$. By tracing out the inner products of $g_{P,\ell}$ with reachability vectors along an accepting path for P, we will be able to come up with a complex state specific to P. To find the complex state, we make use of the following arithmetic lemma.

Lemma 3.3. *Let ℓ, m be integers, and let $x_{t,i}$ be real numbers, where $0 \le t \le \ell$ and $1 \le i \le m$. Suppose that for all $t < \ell$ there exists i such that $x_{t,i} = x_{t+1,i}$. Then*

$$\max_{t,i,j} |x_{t,i} - x_{t,j}| \ge \frac{1}{2\ell} \max_i |x_{\ell,i} - x_{0,i}|.$$

Proof. Let $i = \operatorname{argmax}_i |x_{\ell,i} - x_{0,i}|$ and $\Delta = |x_{\ell,i} - x_{0,i}|$. Since

$$|x_{\ell,i} - x_{0,i}| \le \sum_{t=1}^{\ell} |x_{t,i} - x_{t-1,i}|,$$

there exists $t > 0$ such that $|x_{t,i} - x_{t-1,i}| \ge \Delta/\ell$. Let j be an index such that $x_{t,j} = x_{t-1,j}$. Then

$$\frac{\Delta}{\ell} \le |x_{t,i} - x_{t-1,i}| \le |x_{t,i} - x_{t,j}| + |x_{t-1,j} - x_{t-1,i}|,$$

and so for some $s \in \{t-1, t\}$, $|x_{s,i} - x_{s,j}| \ge \Delta/2\ell$. $\square$

Lemma 3.4 (Gap Lemma)**.** *Let P be any accepting instance of* GEN*, and let $\{g_\ell\}_{\ell \in P}$ be a collection of vectors indexed by literals in P such that for each $\ell \in P$ the corresponding vector g_ℓ is ℓ-nice. Let $\mathbf{M}$ be any sound monotone switching network computing* GEN *with n states, let $\{R_{\mathbf{u}}\}_{\mathbf{u} \in \mathbf{M}}$ be the set of reachability vectors for $\mathbf{M}$, and let W be an $\mathbf{s}$ to $\mathbf{t}$ path in $\mathbf{M}$ which accepts P. Then there is a node $\mathbf{u}$*

on W and two literals $\ell_1, \ell_2 \in P$ for which

$$|\langle R_{\mathbf{u}}, g_{\ell_1} - g_{\ell_2} \rangle| \geq \frac{1}{2n}.$$

Proof. Denote the nodes on W by $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$, where $\mathbf{u}_1 = \mathbf{s}$ and $\mathbf{u}_m = \mathbf{t}$. Let $x_{t,\ell} = \langle R_{\mathbf{u}_t}, g_\ell \rangle$. Lemma 3.2 implies that for each $t < m$ there exists ℓ such that $x_{t,\ell} = x_{t+1,\ell}$, namely the literal ℓ that labels the edge connecting $\mathbf{u}_t$ and $\mathbf{u}_{t+1}$. Apply Lemma 3.3 to get a node $\mathbf{u}$ and two literals ℓ_1, ℓ_2 such that

$$|\langle R_{\mathbf{u}}, g_{\ell_1} \rangle - \langle R_{\mathbf{u}}, g_{\ell_2} \rangle| \geq \frac{1}{2m} \max_\ell |\langle R_{\mathbf{s}}, g_\ell \rangle - \langle R_{\mathbf{t}}, g_\ell \rangle|.$$

Since $R_{\mathbf{s}} \equiv 1$ and $R_{\mathbf{t}} \equiv 0$,

$$|\langle R_{\mathbf{u}}, g_{\ell_1} - g_{\ell_2} \rangle| \geq \frac{1}{2m} \max_\ell |\langle \mathbf{1}, g_\ell \rangle - \langle \mathbf{0}, g_\ell \rangle| = \frac{1}{2m} \geq \frac{1}{2n}. \qquad \square$$

The rest of this section is devoted to constructing the nice vectors which underlie our lower bounds. We will prove the following theorem.

Theorem 3.5. *Let m and h be positive integers. Let P be a graph instance of* GEN *with vertex set V_P isomorphic to a graph G with m vertices and reversible pebbling number at least h. There exist cut vectors $g_{P,\ell}$ for each $\ell \in P$ with the following properties:*

1. *For any $\ell \in P$, $\langle g_{P,\ell}, \mathbf{1} \rangle = 1$.*
2. *For any $\ell \in P$, $g_{P,\ell}$ is ℓ-nice.*
3. *For any $\ell \in P$, $g_{P,\ell}$ depends only on vertices in V_P.*
4. *For any $\ell \in P$, $\|g_{P,\ell}\|^2 \leq (9m)^{h+1}$.*
5. *For any $\ell_1, \ell_2 \in P$ and $S \in \mathcal{C}$ of size $|S| \leq h - 2$, $\hat{g}_{P,\ell_1}(S) = \hat{g}_{P,\ell_2}(S)$.*

As explained in the proof outline, the construction proceeds in two steps. In the first step, we construct vectors $f_{P,\ell}$ satisfying all the required properties other than the bound on the norm. In the second step we "trim" the vectors $f_{P,\ell}$ (by intelligently applying a low-pass filter) to vectors $g_{P,\ell}$ which satisfy all the required properties.

The difficult part of the construction is to control the Fourier spectrum of $g_{P,\ell}$ while guaranteeing that $g_{P,\ell}$ is ℓ-nice. In order to control the Fourier spectrum of $g_{P,\ell}$, we will construct $g_{P,\ell}$ as a linear combination of vectors S_C with known Fourier spectrum. In order to guarantee that $g_{P,\ell}$ is ℓ-nice, we will find an equivalent condition which involves inner products with a different set of vectors K_D. The two sets of vectors S_C, K_D are related by the identity

$$\langle S_C, K_D \rangle = [C = D].$$

(In linear algebra parlance, $\{S_C\}$ and $\{K_D\}$ are dual bases.) Since $g_{P,\ell}$ is a linear combination of the vectors S_C, we will be able to verify that $g_{P,\ell}$ is ℓ-nice using the equivalent condition.

The construction of these S and K vectors will be performed using a particular switching network for graph instances of GEN, which we call the *universal pebbling network*. We think of the universal pebbling network as executing a pebbling algorithm that is highly specific to instances of GEN with a particular underlying graph. The K vectors discussed above have a natural relation with the pebbling configurations appearing in this highly specific pebbling algorithm.

For the rest of this section, we fix a graph instance P whose underlying vertex set is V_P. All cut vectors we define will depend only on vertices in V_P, and so we think of them as real functions on the set $\mathcal{C}_P$ which consists of all subsets of V_P (recall that $s, t \notin V_P$). A vector f defined on $\mathcal{C}_P$ corresponds to the cut vector f' given by

$$f'(C) = f(C \cap V_P).$$

Henceforth, *cut vector* will simply mean a vector defined on $\mathcal{C}_P$. We endow these cut vectors with the inner product

$$\langle f, g \rangle = \frac{1}{|\mathcal{C}_P|} \sum_{C \in \mathcal{C}_P} f(C) g(C).$$

It is not difficult to check that $\|f'\| = \|f\|$, $\langle f', g' \rangle = \langle f, g \rangle$ and that for $C \subseteq V_P$, $\hat{f'}(C) = \hat{f}(C)$. (All other Fourier coefficients of f' vanish since f' depends only on V_P.)

3.2.1 Universal Pebbling Networks

We start by describing the cut vectors K_D and the condition involving them which is equivalent to being ℓ-nice. The cut vectors K_D are defined by

$$K_D(C) = [D \subseteq C].$$

We think of D as a pebbling configuration on the graph G, and of K_D as a kind of reachability vector. To make this identification more precise, we define a monotone switching network related to the reversible pebbling game on G.

Definition 3.6. The *universal pebbling network* for P is a monotone switching network $\mathbf{M}_P$ defined as follows. With every subset of vertices $D \subseteq V_P \cup \{t\}$ we associate a state $\mathbf{D}$ in $\mathbf{M}_P$ and we think of the vertices in $D \cup \{s\}$ as the "pebbled vertices at $\mathbf{D}$". The start state of the network is $\mathbf{s} = \emptyset$, and we add a dummy target state $\mathbf{t}$.

Given two states $\mathbf{D_1}$, $\mathbf{D_2}$ in $\mathbf{M}_P$ and a literal $\ell = (x, y, z)$, we connect $\mathbf{D_1}$ and $\mathbf{D_2}$ with an ℓ-labelled wire if the corresponding pebble configurations $D_1 \cup \{s\}$ and $D_2 \cup \{s\}$ differ by a single legal pebbling move which either pebbles or unpebbles the vertex z. Additionally, for any pebbling configuration D which contains the target point t, we connect the corresponding state $\mathbf{D}$ to the target state $\mathbf{t}$ by a blank wire (i.e. one that is always alive).

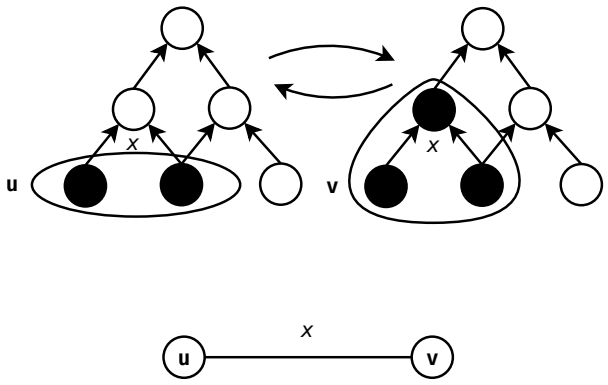

Figure 3.1: Adjacent pebbling configurations and universal switching network states

Figure 3.1 shows two adjacent pebble configurations on the pyramid and the corresponding con-

nections in the universal pebbling network. The pebbling configurations $\mathbf{u}$ and $\mathbf{v}$ on the pyramid have corresponding states in the network, and since they are reachable from one another using the literal x we connect the corresponding states in the universal pebbling network.

The vectors K_D are not quite reachability vectors[3], but they satisfy the following crucial property which is enjoyed by reachability vectors.

Lemma 3.7. *Suppose two states $\mathbf{D_1}, \mathbf{D_2}$ of the universal switching network for P are connected by a wire labelled ℓ. For every cut C such that $\ell \in G(C)$, $K_{D_1}(C) = K_{D_2}(C)$.*

Proof. Let $\ell = (x, y, z)$, and suppose that $D_2 = D_1 \cup \{z\}$. Suppose C is a cut such that $\ell \in G(C)$. If $K_{D_2}(C) = 1$ then clearly $K_{D_1}(C) = 1$, since $D_2 \supseteq D_1$. If $K_{D_1}(C) = 1$ then $x, y \in C_s$. Since $\ell \in G(C)$, this implies that $z \in C_s$, and so $K_{D_2}(C) = 1$. □

We are now ready to state the property involving K_C which is equivalent to being ℓ-nice.

Lemma 3.8. *Let $\ell \in P$ be any literal and let g be any cut vector. If $\langle K_{D_1}, g\rangle = \langle K_{D_2}, g\rangle$ for any two states $\mathbf{D_1}, \mathbf{D_2}$ connected by a wire labelled ℓ then g is ℓ-nice, and the converse also holds.*

Before proving the lemma, we comment that up to the fact that K_D is not quite the reachability vector corresponding to $\mathbf{D}$, the property given by the lemma is the same as being ℓ-nice for the universal switching network.

Proof. We start by proving the converse. Let $\ell \in P$, and suppose that g is ℓ-nice. Let $\mathbf{D_1}, \mathbf{D_2}$ be two states connected by a wire labelled ℓ. Lemma 3.7 implies that

$$\langle K_{D_1}, g\rangle = \sum_{\substack{C \in \mathcal{C}_P \\ \ell \in G(C)}} K_{D_1}(C)g(C) = \sum_{\substack{C \in \mathcal{C}_P \\ \ell \in G(C)}} K_{D_2}(C)g(C) = \langle K_{D_2}, g\rangle,$$

where the first and last equality follows from the fact that g is ℓ-nice.

The other direction is more involved. Let $\ell = (x, y, z)$, and let D be any cut such that $\ell \notin G(D)$, that is $x, y \in D_s$ and $z \notin D_s$. The two states $\mathbf{D}$ and $\mathbf{D} \cup \{\mathbf{z}\}$ are connected by a wire labelled ℓ, and so

[3]For example, let P be a pyramid with vertices $v_{31}, v_{32}, v_{31}, v_{21}, v_{22}, v_{11}$, and let $C = \{v_{31}, v_{32}, v_{33}, v_{11}\}$. The graph $G(C)$ doesn't contain the literals (v_{31}, v_{32}, v_{21}) and (v_{32}, v_{33}, v_{22}), and so the pebbling configuration C isn't reachable. Yet $K_C(C) = 1$.

by the assumption of our lemma we have that

$$0 = |\mathcal{C}_P|\langle K_D - K_{D\cup\{z\}}, g\rangle = \sum_C ([D \subseteq C] - [D \cup \{z\} \subseteq C])g(C) = \sum_{\substack{C \supseteq D \\ z \notin C}} g(C). \tag{3.1}$$

We use (3.1) along with reverse induction on $|D|$ to show that $g(D) = 0$ whenever $\ell \notin G(D)$. Given D, suppose that $g(C) = 0$ for all $C \supsetneq D$ satisfying $\ell \notin G(C)$, or equivalently, $z \notin C$. Equation (3.1) implies that

$$0 = \sum_{\substack{C \supseteq D \\ z \notin C}} g(C) = g(D) + \sum_{\substack{C \supsetneq D \\ z \notin C}} g(C) = g(D). \qquad \square$$

3.2.2 Dual Basis

In view of Lemma 3.8, we will be interested in the value of $\langle g_{P,\ell}, K_C\rangle$ for the function $g_{P,\ell}$ which we will construct. If $\mathcal{K} = \{K_C \mid C \in \mathcal{C}_P\}$ were an orthonormal basis like the Fourier basis, then $\langle g_{P,\ell}, K_C\rangle$ would be the coefficient of K_C in the unique representation of $g_{P,\ell}$ in the basis $\mathcal{K}$. However, the vectors K_C are not orthogonal. We therefore construct another basis $\mathcal{S} = \{S_C \mid C \in \mathcal{C}_P\}$ with the following property:

$$\langle K_C, S_D\rangle = [C = D].$$

The basis $\mathcal{S}$ is known as the *dual basis* to $\mathcal{K}$. This property implies that $\langle g_{P,\ell}, K_C\rangle$ equals the coefficient of S_C in the unique representation of $g_{P,\ell}$ in the basis $\mathcal{S}$: if

$$g_{P,\ell} = \sum_{C \in \mathcal{C}_P} \alpha_C S_C$$

then $\langle g_{P,\ell}, K_C\rangle = \alpha_C$. The dual basis is given by the following formula:

$$S_D(C) = [C \subseteq D] \cdot |\mathcal{C}_P|(-1)^{|D\setminus C|}.$$

We first prove that the two bases are in fact dual.

Lemma 3.9. *For any $C, D \in \mathcal{C}_P$, $\langle K_C, S_D\rangle = [C = D]$.*

Proof. We have

$$\langle K_C, S_D \rangle = \sum_{E \in \mathcal{C}_P} [C \subseteq E][E \subseteq D](-1)^{|D \setminus E|} = \sum_{C \subseteq E \subseteq D} (-1)^{|D \setminus E|}.$$

If C is not a subset of D then clearly $\langle K_C, S_D \rangle = 0$. If $C \subseteq D$ then

$$\langle K_C, S_D \rangle = \sum_{E \subseteq D \setminus C} (-1)^{|D|-|C|-|E|} = [C = D]. \qquad \square$$

The other crucial property of the dual basis is its Fourier expansion.

Lemma 3.10. *For any $C, D \in \mathcal{C}_P$, $\hat{S}_C(D) = [C \subseteq D](-2)^{|C|}$.*

Proof. We have

$$\hat{S}_C(D) = \langle S_C, \chi_D \rangle = \sum_{E \in \mathcal{C}_P} [E \subseteq C](-1)^{|C \setminus E|}(-1)^{|D \cap E|} = \sum_{E \subseteq C} (-1)^{|C \setminus E| + |D \cap E|}.$$

Let $C = \{c_1, \ldots, c_k\}$. Breaking up the sum over C, we have

$$\hat{S}_C(D) = \sum_{E_1 \subseteq \{c_1\}} (-1)^{[c_1 \notin E_1] + [c_1 \in E_1 \cap D]} \cdots \sum_{E_k \subseteq \{c_k\}} (-1)^{[c_k \notin E_k] + [c_k \in E_k \cap D]}.$$

By considering each sum separately, we see that $\hat{S}_C(D) = 0$ unless $C \subseteq D$. When $C \subseteq D$,

$$\begin{aligned} \hat{S}_C(D) &= \sum_{E_1 \subseteq \{c_1\}} (-1)^{[c_1 \notin E_1] + [c_1 \in E_1]} \cdots \sum_{E_k \subseteq \{c_k\}} (-1)^{[c_k \notin E_k] + [c_k \in E_k]} \\ &= \sum_{E_1 \subseteq \{c_1\}} (-1) \cdots \sum_{E_k \subseteq \{c_k\}} (-1) = (-2)^{|C|}. \end{aligned} \qquad \square$$

3.2.3 Nice Vector Construction

Let $\ell = (x, y, z) \in P$. Our plan is to construct $g_{P,\ell}$ as a linear combination of the form

$$g_{P,\ell} = \sum_{C \in \mathcal{C}_P} \alpha_{C,\ell} S_C.$$

Since $K_\emptyset = \mathbf{1}$, the property $\langle g_{P,\ell}, \mathbf{1}\rangle = 1$ implies that $\alpha_{\emptyset,\ell} = 1$. Lemma 3.8 shows that for $g_{P,\ell}$ to be ℓ-nice, we must have $\alpha_{D,\ell} = \alpha_{D\cup\{z\},\ell}$ whenever $x, y \in D_s$ (recall $D_s = D\cup\{s\}$). Finally, Lemma 3.10 shows that for $D \in \mathcal{C}_P$ and any two $\ell_1, \ell_2 \in P$,

$$\hat{g}_{P,\ell_1}(D) - \hat{g}_{P,\ell_2}(D) = \sum_{C\in\mathcal{C}_P} (\alpha_{C,\ell_1} - \alpha_{C,\ell_2})[C \subseteq D]2^{|C|}.$$

Since we want all small Fourier coefficients to be the same for different ℓ, we need $\alpha_{C,\ell_1} = \alpha_{C,\ell_2}$ for all small cuts C.

We can satisfy all the required constraints by setting all $\alpha_{C,\ell}$ to binary values: for each ℓ we will construct a set $A_\ell \subseteq \mathcal{C}_P$ containing $\emptyset$ and define $\alpha_{C,\ell} = [C \in A_\ell]$, or equivalently

$$g_{P,\ell} = \sum_{C\in A_\ell} S_C.$$

Lemma 3.8 shows that for $g_{P,\ell}$ to be ℓ-nice, the set A_ℓ, as a set of pebbling configurations, must be closed under legal pebbling of z. Lemma 3.10 shows that for $D \in \mathcal{C}_P$ and any two $\ell_1, \ell_2 \in P$,

$$\hat{g}_{P,\ell_1}(D) - \hat{g}_{P,\ell_2}(D) = \sum_{C\in\mathcal{C}_P} ([C \in A_{\ell_1}] - [C \in A_{\ell_2}])[C \subseteq D]2^{|C|}.$$

Since we want all small Fourier coefficients to be the same for different ℓ, we need $A_{\ell_1} \triangle A_{\ell_2}$ to contain only large cuts.

When $z = t$, the set $A_t := A_\ell$ must be closed under legal pebbling of t. Since no cuts in $\mathcal{C}_P$ contain t, this implies that in all pebbling configurations in A_t, the vertex t cannot be pebbled. We will enforce this by requiring that each $C \in A_t$ be reachable from $\emptyset$ by using at most $h-2$ pebbles. Since G has reversible pebbling number h, that ensures that t cannot be pebbled. This leads to the following construction, which constructs the preliminary version $f_{P,\ell}$ of $g_{P,\ell}$.

Lemma 3.11. *Let A_t be the set of pebbling configurations reachable from $\emptyset$ using at most $h-2$ pebbles. We think of A_t as a set of states in the universal switching network. For each literal $\ell = (x, y, z)$, let A_ℓ be the closure of A_t under wires labelled ℓ (that is, A_ℓ contains A_t as well as any pebbling configuration reachable from A_t by a wire labelled ℓ). Since the reversible pebbling number of G is h, no pebbling configuration in A_ℓ pebbles t, and so $A_\ell \subseteq \mathcal{C}_P$.*

Define

$$f_{P,\ell} = \sum_{C \in A_\ell} S_C.$$

The functions $f_{P,\ell}$ satisfy the following properties:

1. *For any $\ell \in P$, $\langle f_{P,\ell}, \mathbf{1} \rangle = 1$.*
2. *For any $\ell \in P$, $f_{P,\ell}$ is ℓ-nice.*
3. *For any $\ell_1, \ell_2 \in P$ and $D \in \mathcal{C}_P$ of size $|D| \leq h-2$, $\hat{f}_{P,\ell_1}(D) = \hat{f}_{P,\ell_2}(D)$.*
4. *For any $\ell \in P$ and $D \in \mathcal{C}_P$, $\hat{f}_{P,\ell}(D)^2 \leq 9^{|D|}$.*

Proof. Since $K_\emptyset = \mathbf{1}$,

$$\langle f_{P,\ell}, \mathbf{1} \rangle = \langle f_{P,\ell}, K_\emptyset \rangle = [\{s\} \in A_\ell] = 1.$$

To show that $f_{P,\ell}$ is ℓ-nice, let $\mathbf{D_1}, \mathbf{D_2}$ be two states of the reversible pebbling network connected by a wire labelled ℓ. We have

$$\langle f_{P,\ell}, K_{D_1} \rangle = [D_1 \in A_\ell], \quad \langle f_{P,\ell}, K_{D_2} \rangle = [D_2 \in A_\ell].$$

Since A_ℓ is closed under wires labelled ℓ, $D_1 \in A_\ell$ if and only if $D_2 \in A_\ell$, and so $\langle f_{P,\ell}, K_{D_1} \rangle = \langle f_{P,\ell}, K_{D_2} \rangle$. Lemma 3.8 implies that $f_{P,\ell}$ is ℓ-nice.

For the third property, let $\ell_1, \ell_2 \in P$. We have

$$\hat{f}_{P,\ell_1}(D) - \hat{f}_{P,\ell_2}(D) = \sum_{C \in \mathcal{C}_P} ([C \in A_{\ell_1}] - [C \in A_{\ell_2}])[C \subseteq D] 2^{|C|}.$$

All pebbling configurations in $A_{\ell_1} \triangle A_{\ell_2}$ must contain exactly $h-1$ pebbles, and so $[C \in A_{\ell_1}] \neq [C \in A_{\ell_2}]$ implies $|C| = h-1$. If $|D| \leq h-2$ then no subset of D satisfies this condition, and so $\hat{f}_{P,\ell_1}(D) = \hat{f}_{P,\ell_2}(D)$.

Finally, let $D \in \mathcal{C}_P$. Lemma 3.10 implies that

$$\hat{f}_{P,\ell}(D) = \sum_{C \in A_\ell} [C \subseteq D](-2)^{|C|} \leq \sum_{k=0}^{|D|} \binom{m}{k} 2^k = 3^{|D|}.$$

Similarly we get $\hat{f}_{P,\ell}(D) \geq -3^k$, implying $\hat{f}_{P,\ell}(D)^2 \leq 9^{|D|}$. □

There are at most roughly m^h pebbling configurations in each A_ℓ, and the norm of each S_C is $|\mathcal{C}_P|2^{|C|/2} \leq 2^{m+h/2}$. Therefore we expect the norm of each $f_{P,\ell}$ to be roughly $2^m(\sqrt{2}m)^h$, which is too high. In the next section, we fix the situation by applying a low-pass filter to $f_{P,\ell}$.

3.2.4 Trimming the Nice Vectors

The vectors $f_{P,\ell}$ satisfy all the required properties other than having a small norm. In this section we rectify this situation by relating the Fourier expansion of $f_{P,\ell}$ to the property of being ℓ-nice. We will show that a function is ℓ-nice if its Fourier expansion satisfies certain homogeneous linear equations. If $\ell = (x, y, z)$ then each equation involves Fourier coefficients $C \cup X$ for $X \subseteq \{x, y, z\}$. If we remove the high Fourier coefficients of $f_{P,\ell}$ in a way which either preserves or removes all coefficients of the form $C \cup X$, then we preserve the property of being ℓ-nice while significantly reducing the norm. We need to be careful to maintain the property that the small Fourier coefficients are the same for all $f_{P,\ell}$.

We start by expressing the property of being ℓ-nice in terms of the Fourier coefficients of a cut vector f. For a literal ℓ, define the cut vector

$$I_\ell(C) = [\ell \notin G(C)] = [x \in C_s][y \in C_s][z \notin C_s].$$

A function f is ℓ-nice if for every C, either $\ell \in G(C)$ or $f(C) = 0$. In other words, either $I_\ell(C) = 0$ or $f(C) = 0$, that is to say $I_\ell f \equiv 0$. In order to express this condition as a condition on the Fourier coefficients of f, we use the well-known convolution property of the Fourier transform.

Lemma 3.12. *Let f, g be cut vectors. Define another cut vector $f * g$ by*

$$(f * g)(C) = \sum_{D \subseteq V_P} f(D)g(D \triangle C).$$

Then $\widehat{f * g}(C) = |\mathcal{C}_P|\hat{f}(C)\hat{g}(C)$ and $\widehat{fg}(C) = \hat{f}(C) * \hat{g}(C)$.

Since the Fourier expansion of I_ℓ is supported on coefficients which are a subset of $\{x, y, z\}$, the convolution $\hat{I}_\ell * \hat{f}$ has a particularly simple form.

Lemma 3.13. *Let $\ell = (x, y, z) \in P$, and let $A = \{x, y, z\} \setminus \{s, t\}$. For a cut vector f, the property of*

being ℓ-nice is equivalent to a set of homogeneous equations of the form

$$\sum_{X \subseteq A} \alpha_{i,X} \hat{f}(C_i \cup X) = 0 \tag{3.2}$$

for various cuts $C_i \subseteq \overline{A}$. (The cuts C_i can be repeated, and the coefficients $\alpha_{i,X}$ can be different in different equations.)

Proof. Let $I_\ell(C) = [x \in C][y \in C][z \notin C]$. As we remarked above, f is ℓ-nice if and only if $I_\ell f \equiv 0$, which is true if and only if for all $C \in \mathcal{C}_P$, $0 = \hat{I}_\ell f(C) = (\hat{I}_\ell * \hat{f})(C)$ (using the convolution property). Since all cuts C, the set C_s contains s and does not contain t, I_ℓ depends only on A, and so the equation for C reads

$$0 = (\hat{I}_\ell * \hat{f})(C) = \sum_{X \subseteq A} \hat{I}_\ell(X) \hat{f}(X \triangle C).$$

Write $C = D_1 \triangle D_2$, where $D_1 = C \setminus A$ and $D_2 = C \cap A$. Then the equation for C is equivalent to

$$0 = \sum_{X \subseteq A} \hat{I}_\ell(X) \hat{f}((D_1 \triangle D_2) \triangle X),$$

and since $D_2 \subseteq A$ and we are summing over all $X \subseteq A$ we have

$$\sum_{X \subseteq A} \hat{I}_\ell(X) \hat{f}((D_1 \triangle D_2) \triangle X) = \sum_{X \subseteq A} \hat{I}_\ell(X \triangle D_2) \hat{f}((D_1 \triangle D_2) \triangle (X \triangle D_2) = \sum_{X \subseteq A} \hat{I}_\ell(X \triangle D_2) \hat{f}(D_1 \cup X),$$

which is of the advertised form. □

Because of the particular form of I_ℓ, one can show that the equations we get really depend only on D_1, but we do not need this fact in the proof.

Lemma 3.13 points the way toward the construction of the cut vectors $g_{P,\ell}$ by removing the high Fourier coefficients of the $f_{P,\ell}$ functions.

Theorem 3.5. *Let m and h be positive integers. Let P be a graph instance of* GEN *with vertex set V_P isomorphic to a graph G with m vertices and reversible pebbling number at least h. There exist cut vectors $g_{P,\ell}$ for each $\ell \in P$ with the following properties:*

1. *For any $\ell \in P$, $\langle g_{P,\ell}, \mathbf{1} \rangle = 1$.*

2. *For any $\ell \in P$, $g_{P,\ell}$ is ℓ-nice.*

3. *For any $\ell \in P$, $g_{P,\ell}$ depends only on vertices in V_P.*

4. *For any $\ell \in P$, $\|g_{P,\ell}\|^2 \leq (9m)^{h+1}$.*

5. *For any $\ell_1, \ell_2 \in P$ and $S \in \mathcal{C}$ of size $|S| \leq h-2$, $\hat{g}_{P,\ell_1}(S) = \hat{g}_{P,\ell_2}(S)$.*

Proof. Let $\ell = \{x, y, z\}$ and $A = \{x, y, z\} \setminus \{s, t\}$. We define the cut vector $g_{P,\ell}$ (as a function on $\mathcal{C}_P$) by

$$g_{P,\ell} = \sum_{\substack{C \in \mathcal{C}_P \\ |C \setminus A| \leq h-2}} \hat{f}_{P,\ell}(C)\chi_C.$$

We proceed to verify the properties of $g_{P,\ell}$ one by one. First,

$$\langle g_{P,\ell}, \mathbf{1} \rangle = \hat{g}_{P,\ell}(\emptyset) = \hat{f}_{P,\ell}(\emptyset) = \langle f_{P,\ell}, \mathbf{1} \rangle = 1.$$

Second, for each $C \subseteq \overline{A}$, either all or none of the Fourier coefficients $\{\hat{f}_{P,\ell}(C \cup X) \mid X \subseteq A\}$ appear in $g_{P,\ell}$, and in both cases the condition given by Lemma 3.13 is maintained, showing that $g_{P,\ell}$ is ℓ-nice since $f_{P,\ell}$ is, by Lemma 3.11.

Third, $g_{P,\ell}$ depends only on V_P by construction.

Fourth, Parseval's identity in conjunction with Lemma 3.11 shows that

$$\|g_{P,\ell}\|^2 = \sum_{C \in \mathcal{C}_P} \hat{g}_{P,\ell}(C)^2 \leq \sum_{\substack{C \in \mathcal{C}_P \\ |C| \leq h+1}} \hat{f}_{P,\ell}(C)^2 \leq \sum_{\substack{C \in \mathcal{C}_P \\ |C| \leq h+1}} 9^{|C|} \leq (9m)^{h+1}.$$

Fifth, for $S \in \mathcal{C}_P$ of size $|S| \leq h-2$ and $\ell_1, \ell_2 \in P$,

$$\hat{g}_{P,\ell_1}(S) = \hat{f}_{P,\ell_1}(S) = \hat{f}_{P,\ell_2}(S) = \hat{g}_{P,\ell_2}(S),$$

using Lemma 3.11 once again. □

3.3 Exponential Lower Bounds for Monotone Switching Networks

Using the vectors from Theorem 3.5 we can now prove an exponential lower bound on monotone switching networks. First, note that the final property shows that all the small Fourier coefficients of $g_{P,\ell_1} - g_{P,\ell_2}$ vanish. To take advantage of this property, we employ a combinatorial design in which any two sets intersect in fewer than h points.

Lemma 3.14 (Trevisan [25])**.** *For any positive integers q, m, h with $h \leq m$, there exist q sets $Q_1, Q_2, \ldots, Q_q \subseteq [N]$, where $N = m^2 e^{1+\ln(q)/h}/h$, such that $|Q_i| = m$ for each i and $|Q_i \cap Q_j| \leq h$ for each $i \neq j$.*

We are ready to put the pieces together to prove an exponential lower bound on the size of monotone switching networks for GEN that are correct on all inputs, which we restate here

Theorem 2.15. *Let N, m, h be positive integers, and let G be a DAG on $m+1$ vertices with in-degree at most 2 and reversible pebbling number at least $h+2$. Any sound monotone switching network for* GEN *which accepts all graph instances isomorphic to G must have at least $\Omega(hN/m^2)^{h/3}/O(m)$ states.*

Proof. Lemma 3.14 gives a design $\{Q_1, \ldots, Q_q\}$ of size $q = (hN/em^2)^h$ in which $|Q_i| = m$ and $|Q_i \cap Q_j| \leq h$ for all $i \neq j$. For each Q_i, choose some graph instance P_i isomorphic to G whose underlying vertex set is Q_i. Apply Theorem 3.5 to each instance P_i to get a collection $\{g_{P_i,\ell}\}_{\ell \in P_i}$ of cut vectors.

Let $\mathbf{M}$ be a sound monotone switching network for GEN of size n which accepts all graph instances isomorphic to G. We apply the gap lemma (Lemma 3.4) to each collection of vectors, which gives a set of vectors $\{g_{P_i}\}_{i=1}^q$ and a collection of states $\{\mathbf{u}_i\}_{i=1}^q$ such that $\langle g_{P_i}, \mathbf{u}_i \rangle \geq 1/2n$ and $g_{P_i} = g_{P_i,\ell_1} - g_{P_i,\ell_2}$ for some pair of literals $\ell_1, \ell_2 \in P_i$.

The third property in Theorem 3.5 shows that g_{P_i} depends only on vertices in Q_i, and the final property guarantees that $\hat{g}_{P,i}(C) = 0$ for all $|C| \leq h$. It follows that for $i \neq j$, the functions g_{P_i}, g_{P_j} are orthogonal: if $\hat{g}_{P_i}(C), \hat{g}_{P_j}(C) \neq 0$ then $|C| \geq h+1$ and $C \subseteq Q_i, Q_j$, contradicting the fact that $|Q_i \cap Q_j| \leq h$. Finally, since $\|g_{P_i,\ell}\| \leq \sqrt{(9m)^{h+3}}$ (we get $h+3$ instead of $h+1$ since the reversible pebbling number is at least $h+2$), we get $\|g_{P_i}\| \leq 2\sqrt{(9m)^{h+3}}$.

Since the set $\{g_{P_i}/\|g_{P_i}\| \mid 1 \leq i \leq q\}$ is orthonormal, Parseval's theorem implies that

$$n = \sum_{\mathbf{u} \in \mathbf{M}} 1 \geq \sum_{\mathbf{u} \in \mathbf{M}} \|R_{\mathbf{u}}\|^2 \geq \sum_{\mathbf{u} \in \mathbf{M}} \sum_{i=1}^{q} \left(\frac{\langle R_{\mathbf{u}}, g_{P_i} \rangle}{\|g_{P_i}\|} \right)^2 \geq q \cdot \frac{1}{4n^2} \frac{1}{4(9m)^{h+3}}.$$

We deduce that

$$n^3 \geq \frac{q}{16 \cdot (9m)^{h+3}} \geq \left(\frac{hN}{9em^2}\right)^h (27m)^{-3}. \qquad \square$$

Chapter 4

Average Case Lower Bounds

A natural way to extend the above results is to ask whether or not we can also get a superpolynomial lower bound for randomized monotone switching networks or — by Yao's Minimax Theorem [26] — deterministic monotone switching networks which are allowed to make errors on a distribution of inputs. We will measure the error rate of monotone switching networks in the following way.

If x is an instance of GEN, then we will write $\mathsf{GEN}(x) = 1$ to denote that x is an accepting instance, and $\mathsf{GEN}(x) = 0$ to denote that x is a rejecting instance.

Definition 4.1. Let $\mathcal{D}$ be a distribution on GEN inputs, and let $0 \leq \epsilon < 1/2$ be a real number. We say that a monotone switching network $\mathbf{M}$ *computes* GEN *with* ϵ *error on* $\mathcal{D}$ if $\Pr_{x \sim \mathcal{D}}[\mathbf{M}(x) \neq \mathsf{GEN}(x)] = \epsilon$.

For the rest of the section fix a DAG G on $m+1$ vertices with a unique sink t, in-degree at most 2, and reversible pebbling number at least $h+2$. We will use the following distribution $\mathcal{D}$ over instances of GEN: with probability $1/2$, choose a random graph instance isomorphic to G, and with probability $1/2$ choose $G(C)$ for a random cut C.

Our main result is Theorem 2.17, restated here.

Theorem 2.17. *Let α be a real number in the range $0 < \alpha < 1$. Let m, h, N be positive integers satisfying $324m^2 \leq N^{\alpha}$, and let G be a DAG with $m+1$ vertices, in-degree 2 and reversible pebbling number at least $h+2$.*

Any monotone switching network which computes GEN on $[N+2]$ with error $\epsilon \leq 1/2 - 1/N^{1-\alpha}$ must have at least $\Omega(hN/m^2)^{h/3}/O(mN)$ states.

Examining the proof of the lower bound for error-free switching networks solving GEN reveals two assumptions that fail in the case of switching networks with errors. The first failing assumption is in the beginning of the proof of Theorem 2.15, whereupon choosing a combinatorial design according to Lemma 3.14, we need to identify each of the sets $P_1, \ldots, P_q$ in the design with a graph instance isomorphic to G. However, if we chose a fixed design, some (or many, or even all) of the instances in the design could now be rejected by the switching network and we could no longer apply Lemma 3.4.

The second is that Lemma 3.4 no longer holds: in particular, under the distribution $\mathcal{D}$, there may be cuts C which are incorrectly accepted by the switching network under consideration, and so we can no longer guarantee that $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle = 0$ for every cut vector $g_{P,\ell}$. In the worst case, this means that if any cut vector $g_{P,\ell}$ in the collection required by Lemma 3.4 has $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle = 1$ then the lower bound would fail completely.

Luckily we can mitigate both of these problems. If $\mathbf{M}$ is a monotone switching network computing GEN with errors on some distribution, say that a instance P is *good* for $\mathbf{M}$ if P is accepted by the network and $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle \leq 1 - 1/N$, where $R_{\mathbf{t}}$ is reachability vector of the target state $\mathbf{t}$ in $\mathbf{M}$, $\ell \in P$ is any literal, and $g_{P,\ell}$ is the ℓ-nice vector given by Theorem 3.5.

We need to find a way of generating a combinatorial design, like in Lemma 3.14, such that all of the instances appearing in the design are good. To do this we will show that a random instance is good with high probability, and then we can take a random permutation and apply it to each of the sets appearing in some fixed combinatorial design. It follows that in expectation, the permuted combinatorial design will have many good instances, and by the probabilistic method we can fix a design with many good instances and just ignore the instances which are bad.

However, before we discuss the combinatorial design, we first prove the following lemma, which generalizes Lemma 3.4 to monotone switching networks with errors.

Lemma 4.2 (Generalized Gap Lemma)**.** *Let P be an accepting instance of* GEN*, and let $\{g_\ell\}_{\ell \in P}$ be a collection of vectors indexed by literals in P such that for each $\ell \in P$ the corresponding vector g_ℓ is ℓ-nice. Let $\mathbf{M}$ be a monotone switching network for* GEN *with n states. Let $\{R_{\mathbf{u}}\}_{\mathbf{u} \in \mathbf{M}}$ be the set of reachability vectors for $\mathbf{M}$, and let W be an $\mathbf{s}$ to $\mathbf{t}$ path in $\mathbf{M}$ which accepts P. Suppose that for some $\ell \in P$ we have $\langle R_{\mathbf{t}}, g_\ell \rangle \leq \beta$, where $\mathbf{t}$ is the target state of $\mathbf{M}$. Then there is a node $\mathbf{u}$ on W and two*

literals $\ell_1, \ell_2 \in P$ for which

$$|\langle R_{\mathbf{u}}, g_{\ell_1} - g_{\ell_2}\rangle| \geq \frac{|1-\beta|}{2n}.$$

Proof. Once again, denote the nodes on W by $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$, where $\mathbf{u}_1 = \mathbf{s}$ and $\mathbf{u}_m = \mathbf{t}$. Apply Lemma 3.3 with $x_{t,\ell} = \langle R_{\mathbf{u}_t}, g_\ell\rangle$ to obtain a node $\mathbf{u}$ and two literals ℓ_1, ℓ_2 such that

$$|\langle R_{\mathbf{u}}, g_{\ell_1} - g_{\ell_2}\rangle| = |\langle R_{\mathbf{u}}, g_{\ell_1}\rangle - \langle R_{\mathbf{u}}, g_{\ell_2}\rangle| \geq \frac{1}{2m}\max_{\ell} |\langle R_{\mathbf{s}}, g_\ell\rangle - \langle R_{\mathbf{t}}, g_\ell\rangle| \geq \frac{|1-\beta|}{2n}. \quad \square$$

The next lemma shows that if a uniformly random instance is good with high probability, then we can find a block design with "many" good instances.

Lemma 4.3. *Let N, m, h, q be positive integers. Suppose that there are q sets $S_1, S_2, \ldots S_q \subseteq [N]$ such that the following holds:*

1. *$|S_i| = m$ for all $i \in [q]$, and*
2. *$|S_i \cap S_j| \leq h$ for all $i \neq j$.*

Additionally, suppose that at most a fraction ϵ of the sets $\binom{[N]}{m}$ have some property P. Then there exists a collection of $q' = (1-\epsilon)q$ sets $S'_1, S'_2, \ldots, S'_{q'} \subseteq [N]$ for which both properties stated above hold, and additionally none of the sets S'_i has property P.

Proof. Let π be a uniformly random permutation on $[N]$, and for any set $S \subseteq [N]$ let $\pi(S) = \{\pi(x) : x \in S\}$. For each $i \in [q]$, let X_i be the indicator random variable which is 1 if and only if the set $\pi(S_i)$ has property P. Since π is chosen uniformly at random we have $\Pr[X_i = 1] = \epsilon$, and so by linearity of expectation we get

$$\mathop{\mathrm{E}}_{\pi}\left[\sum_{i=1}^{q} X_i\right] = \sum_{i=1}^{q} \mathop{\mathrm{E}}_{\pi}[X_i] = \sum_{i=1}^{q} \Pr_{\pi}[X_i = 1] = \epsilon q.$$

By the probabilistic method it follows that there must exist q sets $U'_1, \ldots, U'_q$, such that at most ϵq of the sets have property P. Therefore, there exist $q' = (1-\epsilon)q$ sets $S'_1, \ldots, S'_{q'}$ such that both properties in the statement of the lemma hold and none of the sets have property P. $\square$

The actual result we will need is slightly different, but the same proof will work.

Our goal is to upper-bound the probability that a uniformly random instance is bad, which we do by upper-bounding the expectation of $\langle R_{\mathbf{t}}, f_{P,\ell}\rangle^2$ for a randomly chosen pointed instance (P, ℓ) and

applying Markov's inequality. (A pointed instance (P, ℓ) is a graph instance P isomorphic to G along with a literal $\ell \in P$.)

For the rest of this section, let N be an integer and let $\mathbf{M}$ be a monotone switching network of size n computing GEN on $[N+2]$ with error ϵ. For convenience in this section, we assume that $s = N+1$ and $t = N+2$. Recall that $R_{\mathbf{t}}$ is the reachability vector for the target state of $\mathbf{M}$. For a pointed instance (P, ℓ), $g_{P,\ell}$ is the vector constructed using Theorem 3.5.

The error ϵ results from a combination of two errors: the error ϵ_1 on yes instances, and the error ϵ_2 on no instances. According to the definition of $\mathcal{D}$, we have $\epsilon = (\epsilon_1 + \epsilon_2)/2$. We record this observation.

Lemma 4.4. *Let ϵ_1 be the probability that $\mathbf{M}$ rejects a random pointed instance, and let ϵ_2 be the probability that $\mathbf{M}$ accepts a random cut instance (that is, an instance of the form $G(C)$ for a random cut C). Then $\epsilon = (\epsilon_1 + \epsilon_2)/2$.*

Proof. Follows directly from the definition of $\mathcal{D}$. □

Our proofs crucially rely on the following upper bound on the Fourier coefficients of $g_{P,\ell}$.

Lemma 4.5. *Let (P, ℓ) be a pointed instance with underlying vertex set D, and let $C \in \mathcal{C}$ be any cut. If $C \subseteq D$ then $\hat{g}_{P,\ell}(C)^2 \leq 9^{|C|}$, otherwise $\hat{g}_{P,\ell}(C) = 0$.*

Proof. Since $g_{P,\ell}$ depends only on the vertices D, $\hat{g}_{P,\ell}(C) = 0$ unless $C \subseteq D$, so suppose that $C \subseteq D$. The construction of $\hat{g}_{P,\ell}$ in Theorem 3.5 shows that either $\hat{g}_{P,\ell} = 0$ or $\hat{g}_{P,\ell} = \hat{f}_{P,\ell}$, where $f_{P,\ell}$ is the vector constructed in Lemma 3.11. In the latter case, the upper bound follows from the lemma. □

4.1 Warmup

In this section we introduce our technique by proving an upper bound on $|\operatorname{E}_{(P,\ell)}\langle R_{\mathbf{t}}, g_{P,\ell}\rangle|$. While this result isn't strong enough to prove strong lower bounds for monotone switching networks, it will serve to introduce the ideas subsequently used to prove the actual lower bounds.

Instead of estimating $\operatorname{E}_{(P,\ell)}\langle R_{\mathbf{t}}, g_{P,\ell}\rangle$ directly, we will estimate a sum of many such terms all at once. Using this device we are able to take advantage of the fact that the large Fourier coefficients of $g_{P,\ell}$ appear on larger sets, and larger sets are shared by fewer pointed instances. The sum we are going to consider includes a pointed instance (P, ℓ) for each subset Q of $[N]$ of size m (recall $s, t \notin [N]$ in this section).

Definition 4.6. A *pointed instance function* $\wp$ associates with each set $Q \in \binom{[N]}{m}$ a graph instance P isomorphic to G and a literal $\ell \in P$.

We will compute the expectation of

$$\sum_{Q \in \binom{[N]}{m}} \langle R_{\mathbf{t}}, g_{\wp(Q)} \rangle = \langle R_{\mathbf{t}}, \sum_{Q \in \binom{[N]}{m}} g_{\wp(Q)} \rangle$$

over a random $\wp$ (in fact, we will upper-bound this expression for *every* pointed instance function). This will allow us to upper-bound $\mathrm{E}_{(P,\ell)} \langle R_{\mathbf{t}}, g_{P,\ell} \rangle$ using the following lemma.

Lemma 4.7. *Let $\wp$ be a pointed instance function chosen uniformly at random, and let (P, ℓ) be a random pointed instance. We have*

$$\mathop{\mathrm{E}}_{\wp}[\langle R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \rangle] = \binom{N}{m} \mathop{\mathrm{E}}_{(P,\ell)} [\langle R_{\mathbf{t}}, g_{P,\ell} \rangle].$$

Proof. Linearity of expectation shows that

$$\mathop{\mathrm{E}}_{\wp}\left[\langle R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \rangle\right] = \mathop{\mathrm{E}}_{\wp}\left[\sum_{D \in \binom{[N]}{m}} \langle R_{\mathbf{t}}, g_{\wp(D)} \rangle\right] = \sum_{D \in \binom{[N]}{m}} \mathop{\mathrm{E}}_{\wp}[\langle R_{\mathbf{t}}, g_{\wp(D)} \rangle] = \binom{N}{m} \mathop{\mathrm{E}}_{S,\wp} [\langle R_{\mathbf{t}}, g_{\wp(S)} \rangle],$$

where S is chosen randomly from $\binom{[N]}{m}$. The lemma follows since $\wp(S)$ is simply a random pointed instance. □

Here is the actual upper bound, which works for any pointed instance function $\wp$.

Lemma 4.8. *Suppose $N \geq 18m^2$. Let $\wp$ be any pointed instance function. We have*

$$\langle R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \rangle \leq \sqrt{\epsilon_2 \left(1 + \frac{18m^2}{N}\right)} \binom{N}{m}.$$

Proof. By applying Parseval's Theorem and then Cauchy-Schwarz, we get

$$\begin{aligned}\langle R_{\mathbf{t}}, \sum_{D\in\binom{[N]}{m}} g_{\wp(D)}\rangle^2 &\le \left(\sum_{C\in\mathcal{C}} \hat{R}_{\mathbf{t}}(C)\Big(\sum_{D\in\binom{[N]}{m}} \hat{g}_{\wp(D)}(C)\Big)\right)^2 \\ &\le \left(\sum_{C\in\mathcal{C}} \hat{R}_{\mathbf{t}}(C)^2\right)\left(\sum_{C\in\mathcal{C}}\Big(\sum_{D\in\binom{[N]}{m}} \hat{g}_{\wp(D)}(C)\Big)^2\right).\end{aligned}$$

Recall that the network $\mathbf{M}$ computes GEN with error ϵ_2 on cut instances, and so $R_{\mathbf{t}}(C) = 1$ for $\epsilon_2|\mathcal{C}|$ many cuts C. By Parseval's theorem we have

$$\langle R_{\mathbf{t}}, R_{\mathbf{t}}\rangle = \|R_{\mathbf{t}}\|^2 = \sum_{C\in\mathcal{C}} \hat{R}_{\mathbf{t}}(C)^2.$$

Using this, we have

$$\sum_{C\in\mathcal{C}} \hat{R}_{\mathbf{t}}(C)^2 = \frac{1}{|\mathcal{C}|}\sum_{C\in\mathcal{C}} R_{\mathbf{t}}(C)^2 = \frac{\epsilon_2|\mathcal{C}|}{|\mathcal{C}|} = \epsilon_2,$$

and substituting back yields

$$\langle R_{\mathbf{t}}, \sum_{D\in\binom{[N]}{m}} g_{\wp(D)}\rangle^2 \le \epsilon_2 \sum_{C\in\mathcal{C}}\Big(\sum_{D\in\binom{[N]}{m}} \hat{g}_{\wp(D)}(C)\Big)^2.$$

Lemma 4.5 shows that $\hat{g}_{\wp(D)}(C) = 0$ unless $C \subseteq D$, in which case $\hat{g}_{\wp(D)}(C)^2 \le 9^{|C|}$. Additionally, we have the useful estimates

$$\binom{N}{i} \le N^i \quad \text{and} \quad \binom{N-i}{m-i} \le \binom{N}{m}\frac{m^i}{N^i}.$$

Continuing our bound of $\langle R_{\mathbf{t}}, \sum_{D\in\binom{[N]}{m}} g_{\wp(D)}\rangle$ and letting $i = |C|$ for any set C yields

$$\begin{aligned}\epsilon_2 \sum_{C\in\mathcal{C}}\Big(\sum_{D\in\binom{[N]}{m}} \hat{g}_{\wp(D)}(C)\Big)^2 = \epsilon_2 \sum_{C\in\mathcal{C}}\Big(\sum_{\substack{D\in\binom{[N]}{m}\\ D\supseteq C}} \hat{g}_{\wp(D)}(C)\Big)^2 &\le \epsilon_2 \sum_{i=0}^{m}\binom{N}{i}\binom{N-i}{m-i}^2 9^i \\ &\le \epsilon_2 \sum_{i=0}^{m}\binom{N}{m}^2\frac{N^i m^{2i}}{N^{2i}}9^i \le \epsilon_2\binom{N}{m}^2\sum_{i=0}^{m}\left(\frac{9m^2}{N}\right)^i.\end{aligned}$$

We have $9m^2/N \leq 1/2$ by assumption, and so we can bound this sum by a geometric series like so:

$$\sum_{i=0}^{m} \left(\frac{9m^2}{N}\right)^i \leq 1 + \frac{9m^2}{N} \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i = 1 + \frac{18m^2}{N}.$$

Taking square roots finally yields

$$\langle R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \rangle \leq \sqrt{\epsilon_2 \left(1 + \frac{18m^2}{N}\right)} \binom{N}{m}. \quad \square$$

Corollary 4.9. *Suppose $N \geq 18m^2$. For a random pointed instance (P, ℓ),*

$$\mathop{\mathrm{E}}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle] \leq \sqrt{\epsilon_2 \left(1 + \frac{18m^2}{N}\right)}.$$

Proof. Follows directly from Lemma 4.7. $\square$

In this way we can actually get a bound on $|\mathrm{E}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle]|$. However, this bound isn't helpful for showing that $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle$ is often bounded away from 1. It could be that most of the time, $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle = 1$, and rarely $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle$ attains large negative values. In the following section, we rule out this possibility by obtaining a bound on $\mathrm{E}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle^2]$.

4.2 Tensor Square

We now repeat our calculations, this time bounding $\mathrm{E}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle^2]$ instead of $\mathrm{E}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle]$. Markov's inequality will then imply that $\langle R_{\mathbf{t}}, g_{P,\ell} \rangle$ is bounded away from 1 most of the time.

At first glance it seems that our trick of taking a sum of several different pointed instances fails, since $\langle \cdot, \cdot \rangle^2$ isn't linear. We fix that by taking the tensor square of all parties involved.

The *tensor product* of two cut vectors u and v is the vector $u \otimes v \colon \mathcal{C}^2 \to \mathbb{R}$ defined by $(u \otimes v)(C, D) = u(C)v(D)$. We recall the following useful lemma which connects tensor products to the squares of inner products.

Lemma 4.10. *Let u and v be cut vectors. Then $\langle u \otimes u, v \otimes v \rangle = \langle u, v \rangle^2$, and $\widehat{f \otimes g}(C, D) = \hat{f}(C)\hat{g}(D)$.*

Using tensor products, we are able to extend Lemma 4.7 to a result which is useful for bounding $\mathrm{E}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle^2]$.

Lemma 4.11. *Let $\wp$ be a pointed instance function chosen uniformly at random, and let (P, ℓ) be a random pointed instance. We have*

$$\mathop{\mathrm{E}}_{\wp}[\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \otimes g_{\wp(D)} \rangle] = \binom{N}{m} \mathop{\mathrm{E}}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell} \rangle^2].$$

Proof. As in the proof of Lemma 4.7, we get

$$\mathop{\mathrm{E}}_{\wp}[\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \otimes g_{\wp(D)} \rangle] = \binom{N}{m} \mathop{\mathrm{E}}_{(P,\ell)}[\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, g_{P,\ell} \otimes g_{P,\ell} \rangle].$$

The lemma now follows from Lemma 4.10. □

We proceed with the analog of Lemma 4.8, whose proof is similar in spirit to the proof of Lemma 4.8.

Lemma 4.12. *Suppose $N \geq 162m^2$. Let $\wp$ be any pointed instance function. We have*

$$\langle R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \rangle \leq \sqrt{\epsilon_2 \left(1 + \frac{324m^2}{N}\right)} \binom{N}{m}.$$

Proof. Since the network $\mathbf{M}$ accepts a random cut with probability ϵ_2 we have $\langle R_{\mathbf{t}}, R_{\mathbf{t}} \rangle = \epsilon_2$. We can therefore apply Lemma 4.10 to get

$$\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, R_{\mathbf{t}} \otimes R_{\mathbf{t}} \rangle = \langle R_{\mathbf{t}}, R_{\mathbf{t}} \rangle^2 = \epsilon_2^2.$$

Using this fact, applying Parseval's identity, Cauchy-Schwarz, and Parseval again yields

$$\begin{aligned}\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \otimes g_{\wp(D)} \rangle^2 &\leq \left(\sum_{C_1 \in \mathcal{C}} \sum_{C_2 \in \mathcal{C}} \widehat{(R_{\mathbf{t}} \otimes R_{\mathbf{t}})}(C_1, C_2) \sum_{D \in \binom{[N]}{m}} \widehat{(g_{\wp(D)} \otimes g_{\wp(D)})}(C_1, C_2) \right)^2 \\ &\leq \epsilon_2^2 \sum_{C_1 \in \mathcal{C}} \sum_{C_2 \in \mathcal{C}} \left(\sum_{D \in \binom{[N]}{m}} \widehat{(g_{\wp(D)} \otimes g_{\wp(D)})}(C_1, C_2) \right)^2.\end{aligned}$$

Applying the second conclusion of Lemma 4.10 we get

$$\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \otimes g_{\wp(D)} \rangle^2 \leq \epsilon_2^2 \sum_{C_1 \in \mathcal{C}} \sum_{C_2 \in \mathcal{C}} \left(\sum_{D \in \binom{[N]}{m}} \hat{g}_{\wp(D)}(C_1) \hat{g}_{\wp(D)}(C_2) \right)^2.$$

Recall from Lemma 4.5 that $|\hat{g}_{\wp(D)}(C)| \leq 3^{|C|}$ if $C \subseteq D$, and otherwise $\hat{g}_{\wp(D)}(C) = 0$. We can use this to simplify the sum and get

$$\sum_{C_1 \in \mathcal{C}} \sum_{C_2 \in \mathcal{C}} \left(\sum_{D \in \binom{[N]}{m}} \hat{g}_{\wp(D)}(C_1) \hat{g}_{\wp(D)}(C_2) \right)^2 \leq \sum_{C_1 \in \mathcal{C}} \sum_{C_2 \in \mathcal{C}} \left(\sum_{\substack{D \in \binom{[N]}{m} \\ D \supseteq C_1 \cup C_2}} 3^{|C_1|+|C_2|} \right)^2.$$

Now, for any two cuts $C_1, C_2 \in \mathcal{C}$ appearing in the above sum, let $i = |C_1 \cap C_2|, j = |C_1 \setminus C_2|$ and $k = |C_2 \setminus C_1|$. We can rewrite the sum as

$$\begin{aligned}
&\sum_{C_1 \in \mathcal{C}} \sum_{C_2 \in \mathcal{C}} \left(\sum_{\substack{D \in \binom{[N]}{m} \\ D \supseteq C_1 \cup C_2}} 3^{|C_1|+|C_2|} \right)^2 \\
\leq &\sum_{i=0}^{m} \sum_{j=0}^{m-i} \sum_{k=0}^{m-i-j} \binom{N}{i,j,k} \binom{N-i-j-k}{m-i-j-k}^2 9^{2i+j+k},
\end{aligned}$$

where $\binom{N}{i,j,k}$ is a multinomial coefficient. We can use the upper bound $\binom{N}{i,j,k} \leq N^{i+j+k}$ to get

$$\begin{aligned}
&\sum_{i=0}^{m} \sum_{j=0}^{m-i} \sum_{k=0}^{m-i-j} \binom{N}{i,j,k} \binom{N-i-j-k}{m-i-j-k}^2 9^{2i+j+k} \\
\leq &\sum_{i=0}^{m} \sum_{j=0}^{m-i} \sum_{k=0}^{m-i-j} N^{i+j+k} \binom{N}{m}^2 \frac{m^{2(i+j+k)}}{N^{2(i+j+k)}} 9^{2i+j+k} \\
\leq &\binom{N}{m}^2 \sum_{i=0}^{m} \sum_{j=0}^{m-i} \sum_{k=0}^{m-i-j} \left(\frac{81m^2}{N} \right)^{i+j+k}.
\end{aligned}$$

We can upper-bound this sum by factoring a larger sum:

$$\begin{aligned}
&\binom{N}{m}^2 \sum_{i=0}^{m} \sum_{j=0}^{m-i} \sum_{k=0}^{m-i-j} \left(\frac{81m^2}{N} \right)^{i+j+k} \\
\leq &\binom{N}{m}^2 \sum_{i=0}^{m} \sum_{j=0}^{m} \sum_{k=0}^{m} \left(\frac{81m^2}{N} \right)^{i+j+k} \\
\leq &\binom{N}{m}^2 \left(\sum_{i=0}^{m} \frac{81^i m^{2i}}{N^i} \right) \left(\sum_{j=0}^{m} \frac{81^j m^{2j}}{N^j} \right) \left(\sum_{k=0}^{m} \frac{81^k m^{2k}}{N^k} \right).
\end{aligned}$$

Since $N \geq 162m^2$ by assumption we can upper-bound these using geometric series, as in the proof of

Lemma 4.8:

$$\binom{N}{m}^2 \left(\sum_{i=0}^{m} \frac{81^i m^{2i}}{N^i}\right)\left(\sum_{j=0}^{m} \frac{81^j m^{2j}}{N^j}\right)\left(\sum_{k=0}^{m} \frac{81^k m^{2k}}{N^k}\right) \leq \binom{N}{m}^2 \left(1+\frac{162m^2}{N}\right)^3.$$

Taking square roots, we get

$$\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \otimes g_{\wp(D)} \rangle \leq \epsilon_2 \left(1+\frac{162m^2}{N}\right)^{3/2} \binom{N}{m} \leq \epsilon_2 \left(1+\frac{324m^2}{N}\right) \binom{N}{m}. \quad \square$$

Corollary 4.13. *Suppose $N \geq 162m^2$. For a random pointed instance (P, ℓ),*

$$\mathop{\mathrm{E}}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle^2] \leq \epsilon_2 \left(1+\frac{324m^2}{N}\right).$$

Proof. Follows directly from Lemma 4.11. $\square$

Applying Markov's inequality, we immediately get the following corollary.

Corollary 4.14. *Suppose $N \geq 162m^2$. Let (P, ℓ) be a random pointed instance. For any $\delta > 0$,*

$$\Pr_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle \geq \delta] \leq \delta^{-2} \left(1+\frac{324m^2}{N}\right) \epsilon_2.$$

4.3 Exponential Lower Bounds for Monotone Switching Networks with Errors

We are now ready to prove our main theorem (Theorem 2.17), which gives an exponential lower bound on the size of monotone switching networks computing GEN with error close to $1/2$.

Proof of Theorem 2.17. Lemma 3.14 gives a design $\{Q_1, \ldots, Q_q\}$ of size $q = (hN/em^2)^h$ in which $|Q_i| = m$ and $|Q_i \cap Q_j| \leq h$ for all $i \neq j$. Let π be a random permutation on $[N]$, and let $\wp$ be a random pointed instance function. For each Q_i, $\wp(\pi(Q_i))$ is a random pointed instance, and hence for any δ in the range $0 < \delta < 1$,

$$\Pr_{\pi,\wp}[\langle R_{\mathbf{t}}, g_{\wp(\pi(Q_i))}\rangle \geq \delta] \leq \delta^{-2} \left(1+\frac{324m^2}{N}\right) \epsilon_2.$$

Moreover, the probability that $\wp(\pi(Q_i))$ is rejected by $\mathbf{M}$ is ϵ_1. The probability that either of these bad events happen is at most

$$\tau := \epsilon_1 + \delta^{-2}\left(1 + \frac{324m^2}{N}\right)\epsilon_2 \leq \delta^{-2}\left(1 + \frac{324m^2}{N}\right)2\epsilon,$$

using Lemma 4.4. The technique of Lemma 4.3 shows that for some π and $\wp$, the number of Q_i for which either $\langle R_{\mathbf{t}}, g_{\wp(\pi(Q_i))}\rangle \geq \delta$ or $\wp(\pi(Q_i))$ is rejected by $\mathbf{M}$ is at most τq. In other words, there exists a set $\{(P_1, \ell_1), \ldots, (P_{q^*}, \ell_{q^*})\}$ of $q^* := (1-\tau)q$ pointed instances with underlying sets $Q_1^*, \ldots, Q_{q^*}^*$ such that $|Q_i^* \cap Q_j^*| \leq h$ for all $i \neq j$, each instance P_i is accepted by $\mathbf{M}$, and for any literal $\ell \in P_i$ we have $\langle R_{\mathbf{t}}, g_{P_i,\ell}\rangle \leq \delta$.

The rest of the proof directly follows the proof of Theorem 2.15. We can apply Theorem 3.5 to each instance P_i and get a collection of vectors $\{g_{P_i,\ell}\}_{\ell \in P_i}$, and then apply the generalized gap lemma (Lemma 4.2) to each such collection of vectors, which yields a set of vectors $\{g_{P_i}\}_{i=1}^{q^*}$ and a set of states $\{\mathbf{u}_i\}_{i=1}^{q^*}$ in $\mathbf{M}$ satisfying $\langle g_{P_i}, R_{\mathbf{u}_i}\rangle \geq (1-\delta)/2n$ (recall n is the size of $\mathbf{M}$). This collection of vectors is orthogonal, and each vector satisfies the upper bound $\|g_i\|^2 \leq 4(9m)^{h+1}$. Since the set $\{g_{P_i}/\|g_{P_i}\| : 1 \leq i \leq q\}$ is orthonormal, Parseval's theorem implies that

$$n = \sum_{\mathbf{u}\in\mathbf{M}} 1 \geq \sum_{\mathbf{u}\in\mathbf{M}} \|R_{\mathbf{u}}\|^2 \geq \sum_{\mathbf{u}\in\mathbf{M}}\sum_{i=1}^{q^*}\left(\frac{\langle R_{\mathbf{u}}, g_{P_i}\rangle}{\|g_{P_i}\|}\right)^2 \geq q^* \cdot \frac{(1-\delta)^2}{4n^2}\frac{1}{4(9m)^{h+3}}.$$

We deduce that

$$n^3 \geq \frac{q^*(1-\delta)^2}{16\cdot(9m)^{h+3}} \geq (1-\tau)(1-\delta)^2\left(\frac{hN}{9em^2}\right)^h (27m)^{-3}.$$

An auspicious choice for δ is $\delta = 1 - 1/N$. Since $\epsilon \leq 1/2 - 1/N^{1-\alpha}$ and $324m^2/N \leq N^{\alpha-1}$,

$$\tau \leq \left(1 + \frac{324m^2}{N}\right)2\epsilon \leq \left(1 + \frac{1}{N^{1-\alpha}}\right)\left(1 - \frac{2}{N^{1-\alpha}}\right) \leq 1 - \frac{1}{N^{1-\alpha}}.$$

Therefore

$$n^3 \geq \frac{1}{N^{1-\alpha}}\frac{1}{N^2}\left(\frac{hN}{9em^2}\right)^h (27m)^{-3} \geq \left(\frac{hN}{9em^2}\right)^h (27mN)^{-3}. \qquad \square$$

Bibliography

[1] N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[2] A. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl*, 31(3):530–534, 1985.

[3] P. Beame, T. Pitassi, and N. Segerlind. Lower bounds for Lovász Schrijver from multiparty communication complexity. In *SIAM J. Computing*, 2007.

[4] C. Berg and S. Ulfberg. Symmetric approximation arguments for monotone lower bounds without sunflowers. *Computational Complexity*, 8:1–20, 1999.

[5] A. Bogdanov and L. Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[6] Allan Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, December 1977.

[7] Siu Man Chan. Just a pebble game. In *CCC*, 2013.

[8] Siu Man Chan and Aaron Potechin. Tight bounds for monotone switching networks via fourier analysis. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 495–504. ACM, 2012.

[9] Stephen A. Cook. An observation on time-storage trade off. In *Proceedings of the fifth annual ACM Symposium on Theory of computing*, STOC '73, pages 29–33, New York, NY, USA, 1973. ACM.

[10] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149–161, April 1985.

[11] Yuval Filmus, Toniann Pitassi, Robert Robere, and Stephen Cook. Average case lower bounds for monotone switching networks. Submitted, 2013.

[12] John R. Gilbert and Robert Endre Tarjan. Variations on a pebble game on graphs. Technical Report STAN-CS-78-661, Stanford University, 1978.

[13] M. Grigni and M. Sipser. Monotone separation of logarithmic space from logarithmic depth. *JCSS*, 50:433–437, 1995.

[14] A. Haken. Counting bottlenecks to show monotone $P \neq NP$. In *FOCS*, pages 36–40, 1995.

[15] D. Harnik and R. Raz. Higher lower bounds on monotone size. In *STOC*, pages 378–387, 2000.

[16] T. Huynh and J. Nordstrom. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space tradeoffs in proof complexity. In *44th STOC*, pages 233–248, 2012.

[17] G. Karakostas, J. Kinne, and D. van Melkebeek. On derandomization and average-case complexity of monotone functions. *Theoretical Computer Science*, 434:35–44, 2012.

[18] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 539–550, Chicago, IL, May 1988.

[19] Jakob Nordstöm. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Manuscript.

[20] R. O'Donnell and K. Wimmer. KKL, Kruskal-Katona, and monotone nets. In *FOCS*, pages 725–734, 2009.

[21] A. Potechin. Bounds on monotone switching networks for directed connectivity. In *FOCS*, pages 553–562, 2010.

[22] R. Raz and P. McKenzie. Separation of the monotone NC hierarchy. In *Proceedings of 38th IEEE Foundations of Computer Science*, 1997.

[23] Ran Raz and Avi Wigderson. Probabilistic communication complexity of Boolean relations. In *30th Annual Symposium on Foundations of Computer Science*, pages 562–567, Research Triangle Park, NC, October 1989. IEEE. Full version.

[24] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean function. *Soviet Math. Dokl.*, 31:354–357, 1985.

[25] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[26] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

Appendix A

Relating Monotone Switching Network Size to Monotone Circuit Depth

In this appendix we show that a monotone circuit of depth d can be simulated by a monotone switching network of size 2^d, and a monotone switching network of size s can be simulated by a monotone circuit of depth $O(\log^2 s)$. The corresponding results for the nonmonotone case (with switching networks replaced by branching programs) were proved by Borodin [6].

The second result is easy: Given a monotone switching network of size s together with its input bits $x_1, \ldots, x_n$, we construct an $s \times s$ Boolean matrix A, where $A_{ij} = 1$ iff $i = j$ or there is a wire between state i and state j with a label $x_k = 1$. Now the circuit computes A^s by squaring the matrix $\log s$ times. The network accepts the input iff $A_{ab} = 1$, where a is the initial state and b is the accepting state.

Now we show how a monotone switching network can evaluate a monotone circuit C of depth d. We may assume that C is a balanced binary tree where the root is the output gate (at level 1) and the nodes at levels 1 to d are gates, each labelled either $\wedge$ or $\vee$, and each leaf at level $d + 1$ is labelled with one of the input variables x_i. The idea is to simulate an algorithm which uses a depth first search of the circuit to evaluate its gates.

We assume that the circuit is laid out on the plane with the output gate at the bottom and the input nodes at the top, ordered from left to right. A *full path* is a path in the circuit from the output gate to some input node.

The states of the network consist of a start state $\mathbf{s}$, a target state $\mathbf{t}$, and a state $\mathbf{u_p}$ for each full path p

in the circuit (so there are 2^d such states).

Definition A.1. We say that a path p from a gate g in C to some input node is *initial* if p leaves every $\wedge$-gate g on the path via the left input to g. We say that p is *final* if p leaves every $\wedge$-gate via the right input to g. If p is a full path, then the state $\mathbf{u_p}$ is *initial* if p is initial, and $\mathbf{u_p}$ is *final* if p is final.

We say that a state $\mathbf{u_p}$ is *sound* (for a given setting of the input bits $\vec{x}$) if for each $\wedge$-gate g on the path p, if the path leaves g via its right input, then the left input to g has value 1.

We will construct our network so that following holds.

Claim A.2. For each input $\vec{x}$, a state $\mathbf{u_p}$ is reachable iff $\mathbf{u_p}$ is sound.

Now we define the wires and their labels in the network.

The start state $\mathbf{s}$ is connected via a wire labelled 1 (that is, this wire is always alive) to every initial state $\mathbf{u_p}$. Note that every initial state is (vacuously) sound, as required by the claim.

For every nonfinal full path p to some input x_i, $\mathbf{u_p}$ is connected by a wire labelled x_i to every state $\mathbf{u_{p'}}$ such that p' is a full path which follows p up to the last $\wedge$-gate g such that p leaves g via its left input, and then p' leaves g via its right input and continues along any initial path to a circuit input.

The following is easy to verify:

Claim A.3. It p, p' and x_i are as above, and $x_i = 1$, then $\mathbf{u_p}$ is sound iff $\mathbf{u_{p'}}$ is sound.

Claim A.2 follows from Claim A.3 and the facts that all initial states are sound, and every noninitial sound state $\mathbf{u_{q'}}$ is connected by a wire labelled 1 to a sound state $\mathbf{u_q}$ where the path q is to the left of path q'.

For every final full path p to some input x_i, the state $\mathbf{u_p}$ is connected to the target $\mathbf{t}$ by a wire labelled x_i. Note that if $\mathbf{u_p}$ is reachable and sound, and $x_i = 1$, then every gate along p has value 1, including the output gate. This and Claim A.2 shows that the network is sound (the circuit output is 1 if the target $\mathbf{t}$ in the network is reachable).

Conversely, if the circuit outputs 1, then there is a sound final full path p which witnesses it. Claim A.2 shows that $\mathbf{u_p}$ is reachable, and so $\mathbf{t}$ is reachable. We conclude that the network is complete.

Appendix B

More on Randomized Karchmer-Wigderson

Our work in Section 2.6 implies that the Karchmer-Wigderson reduction fails in the randomized setting. Here is a concrete example. Let $f_\triangle$ be the function whose input is an undirected graph G on n vertices, and $f_\triangle(G) = 1$ if G contains some triangle. The corresponding communication problem is: Alice gets a graph A with a triangle, Bob gets a graph B without a triangle, and they have to come up with an edge e such that $e \in A$ and $e \notin B$. We consider the following probability distributions over yes and no inputs: Alice gets a random triangle, and Bob gets a complete bipartite graph generated by a random partition of $\{1, \ldots, n\}$. Here is a deterministic protocol for $R^m_{f_\triangle}$ with success probability $3/4$:

1. Alice sends Bob the vertices i, j, k of her input triangle (we assume $i < j < k$).
2. If $(i, j) \notin B$ then Bob outputs (i, j), else Bob outputs (i, k).

The Karchmer-Wigderson transformation produces the following formula:

$$\phi = \bigvee_{i<j<k} x_{i,j} \wedge x_{j,k},$$

where $x_{s,t}$ is the input variable corresponding to the edge (s, t). The formula ϕ is true with high probability over a random no input. Intuitively, Alice can cheat: instead of choosing a bona fide triangle i, j, k, she identifies two edges $(i, j), (j, k)$ in the input graph, and pretends as if her input triangle were i, j, k. In order to make this intuition more precise, we digress into the game semantics of monotone

circuits.

Let C be a monotone circuit with input x. We define a combinatorial game $\eth(C)$ between two players, Alice and Bob, whose value depends on the input x. The game starts at the root node of the circuit, and progresses toward the leaves. Alice's goal is to guide the game toward a leaf corresponding to a 1 input bit, and Bob's goal is to guide the game toward a leaf corresponding to a 0 input bit. At a node $v = v_1 \vee \cdots \vee v_k$, Alice decides which of the nodes $v_1, \ldots, v_k$ to go next to. At a node $v = v_1 \wedge \cdots \wedge v_k$, Bob decides which of the nodes $v_1, \ldots, v_k$ to go next to. At a leaf $f = x_i$, the input bit x_i is revealed: if $x_i = 1$ then Alice wins, if $x_i = 0$ then Bob wins. Under optimal play, one of the two players always wins (this is a property of combinatorial games with perfect information). In fact, if $C(x) = 1$ then Alice wins under optimal play, and if $C(x) = 0$ then Bob wins under optimal play. This can be proved by induction on the structure of the circuit.

Having identified monotone circuits with games of the form $\eth(C)$, we describe the Karchmer-Wigderson reduction as a game. Given a function f and a (correct) protocol P for R_f^m, the game $\eth(P)$ proceeds as follows. The states of the game are partial transcripts of the protocol P. We only consider partial transcripts which actually occur for some yes input of Alice and some no input of Bob. The initial state is the empty transcript. At a partial transcript τ, if it's Alice's turn to speak in the protocol, then she chooses whether to proceed to $\tau 0$ or to $\tau 1$. If one of these, say $\tau 0$, never occurs as a partial transcript of the protocol, then she is forced to choose the other, in this case $\tau 1$. When it's Bob's turn to speak, he gets to decide which way to proceed. At a completed transcript τ with output i, the game terminates by revealing the input bit x_i.

The game $\eth(P)$ has the feature that if $f(x) = 1$ then Alice wins, while if $f(x) = 0$ then Bob wins. Since the situation is symmetric, we concentrate on the case that $f(x) = 1$. We show that Alice has a winning strategy for $\eth(P)$. Her winning strategy is simple: at a partial transcript τ when it's her turn to speak, Alice acts according to the protocol P, assuming her input is x. The game terminates at a transcript τ which corresponds to some input y for Bob. Since P is correct, it outputs a bit i such that $x_i = 1$ and $y_i = 0$. Since $x_i = 1$, Alice wins $\eth(P)$.

There is an important difference between the communication protocol P and the corresponding game $\eth(P)$: in the communication protocol P, both players are cooperating to find the bit i which monotonically distinguishes their inputs. In contrast, the game $\eth(P)$ is competitive. If $f(x) = 1$ then Alice wins since she has a winning strategy, in which she truthfully follows the protocol P. Whatever

Bob does is consistent with some input y for him, and since P is correct, at the end the protocol outputs a bit i such that $x_i = 1$.

At this point, we come back to our earlier protocol for $R^m_{F_\triangle}$. The formula ϕ corresponds to the following game:

1. Alice chooses vertices i, j, k.

2. Bob chooses whether to reveal $x_{i,j}$ or $x_{j,k}$.

This game is heavily skewed toward Alice. Suppose x is a bipartite graph which contains two edges $(i, j), (j, k)$. Alice acts as if her input was the triangle $\{i, j, k\}$, and wins the game. Even though for roughly $3/4$ of the triangles, the protocol would succeed (the exact probability depends on x), in the corresponding game Alice always wins, since she gets to choose which triangle to run the protocol against, and she chooses one for which the protocol fails.