Lower Bounds and Derandomization

by

Jaffer Gardezi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Jaffer Gardezi

Abstract

A major open problem in complexity theory is to determine whether randomized complexity classes such as BPP, MA, and AM have any nontrivial derandomization. This thesis investigates the derandomization of two randomized versions of the polynomial hierarchy, AM^{Σ_i} and MA^{Σ_i}. The existence of a nontrivial derandomization of MA^{Σ_i} is shown to be equivalent to a polynomial size Σ_i-oracle circuit lower bound for $\Sigma_{i+1}^{\mathrm{EXP}}$, the i^{th} level of the exponential hierarchy, for $i \geq 0$. This extends an analogous result of Impagliazzo et. al. concerning MA [10]. This equivalence is used to show that such a derandomization exists for all except at most one level of this hierarchy. Concerning AM^{Σ_i}, a tradeoff is derived between derandomizations of this hierarchy and of BPP. Also, it is shown that previously known tradeoffs between derandomizations of AM and deterministic simulations of nondeterministic time generalize to AM^{Σ_i}.

Acknowledgements

I would like to thank my supervisor, Jonathan Buss, and my thesis readers, Alex López-Ortiz and Prabhakar Ragde.

Contents

Chapter 1

Introduction

1.1 Randomized Computation

The goal of complexity theory is to determine the extent to which various resources facilitate computation, and one of the most important such resources is randomness. A randomized algorithm for a problem is one whose computation on a given instance of the problem depends on a series of uniformly generated random bits, and which decides the problem correctly with high probability taken over these bits. Randomization is used extensively in the design of algorithms, as it provides simpler and more efficient algorithms for many problems. For some problems, such as polynomial identity testing, the only known polynomial time algorithms are probabilistic.

In complexity theory, various classes of decision problems that can be solved efficiently using randomness are defined. One of the most important of these classes is BPP, the class of bounded error probabilistic polynomial time algorithms. BPP is defined as the class of decision problems for which there exists a polynomial time randomized algorithm with the following property. The algorithm either accepts or rejects its input with probability taken over all random bit strings at least $\frac{1}{2} + \epsilon$ for some constant error bound ϵ. The algorithm is defined to accept if it accepts for most random bit strings. More formally, a language L is in BPP if, for input x, there is a polynomial time decidable predicate M and an $\epsilon > 0$ such that,

$$\begin{aligned} x \quad &\in L \leftrightarrow \Pr\left[M\left(x, y\right) = 1\right] > \frac{1}{2} + \epsilon \\ x \quad &\notin L \leftrightarrow \Pr\left[M\left(x, y\right) = 1\right] < \frac{1}{2} - \epsilon \end{aligned} \tag{1.1}$$

where the probability is taken over $y \in \{0,1\}^m$ for m polynomial in $|x|$. The importance of BPP arises from the following theorem, which implies

that any problem in BPP can be decided correctly in polynomial time with exponentially low probability of error.

Theorem 1.1 Take $L \in \text{BPP}$ and let q be any polynomial. Then there exists a polynomial time decidable predicate M' such that

$$\begin{aligned} x &\in L \leftrightarrow \Pr\left[M'\left(x,y\right)=1\right] > 1-2^{-q(n)} \\ x &\notin L \leftrightarrow \Pr\left[M'\left(x,y\right)=1\right] < 2^{-q(n)} \end{aligned} \tag{1.2}$$

Proof The following proof is taken from Bovet and Crescenzi [3]. Let A be the probabilistic algorithm deciding L with error bound ϵ. Define a BPP algorithm A' for L as follows. A' runs A t times for some odd t, and accepts if A accepts at least $\frac{t}{2}$ times. The probability that $\text{A}'\left(x\right)$ rejects for $x \in L$ is

$$\begin{aligned} &\sum_{i=0}^{\frac{(t-1)}{2}} \binom{t}{i} \left(\frac{1}{2}+\epsilon\right)^{i} \left(\frac{1}{2}-\epsilon\right)^{t-i} \\ &< \sum_{i=0}^{\frac{(t-1)}{2}} \binom{t}{i} \left(\frac{1}{2}+\epsilon\right)^{i} \left(\frac{1}{2}-\epsilon\right)^{t-i} \left(\frac{\frac{1}{2}+\epsilon}{\frac{1}{2}-\epsilon}\right)^{\frac{t}{2}-i} \\ &= \sum_{i=0}^{\frac{(t-1)}{2}} \binom{t}{i} \left(\frac{1}{2}+\epsilon\right)^{\frac{t}{2}} \left(\frac{1}{2}-\epsilon\right)^{\frac{t}{2}} \\ &= \sum_{i=0}^{\frac{(t-1)}{2}} \binom{t}{i} \left(\frac{1}{4}-\epsilon^{2}\right)^{\frac{t}{2}} \\ &= \frac{1}{2}\left(\frac{1}{4}-\epsilon^{2}\right)^{\frac{t}{2}} \sum_{i=0}^{t} \binom{t}{i} \\ &= \frac{1}{2}\left(\frac{1}{4}-\epsilon^{2}\right)^{\frac{t}{2}} 2^{t} \\ &= \frac{1}{2}\left(1-4\epsilon^{2}\right)^{\frac{t}{2}} \end{aligned} \tag{1.3}$$

Therefore, A' accepts $x \in L$ with probability at least $1-\frac{1}{2}\left(1-4\epsilon^{2}\right)^{\frac{t}{2}}$. For

$$t \geq \frac{2\left(q\left(n\right)-1\right)}{\log\left(\frac{1}{1-4\epsilon^{2}}\right)} \tag{1.4}$$

this probability is greater than $1 - 2^{-q(n)}$. A similar argument shows that, for t iterations with t satisfying (1.4), the probability that A$'$ accepts $x \notin L$ is less than $2^{-q(n)}$. Since t is polynomial in n and A runs in polynomial time, A$'$ is polynomial time.

Other important polynomial time randomized complexity classes are RP, or randomized polynomial time, and ZPP, or zero-error probabilistic polynomial time. These are classes of languages recognized by algorithms satisfying additional restrictions on their behaviour on any given input. A language is in RP if there is an algorithm that accepts words in the language with high probability, and always rejects words not in the language. ZPP is the class of languages for which there is a randomized algorithm that decides its input correctly with high probability, and in all other cases returns the answer "don't know".

In addition to randomized versions of deterministic polynomial time, it is also useful to define randomized extensions of the class NP. NP is the class of languages for which there exist proofs of membership that can be verified in P. The class AM of Arthur-Merlin games is defined in terms of a more elaborate notion of provability: provability via a game between Merlin, an all-powerful prover, and Arthur, a probabilistic verifier. The game is played out in a series of alternating Merlin and Arthur rounds, with Merlin trying to convince Arthur that a word is in a language and Arthur attempting to verify Merlin's proof. Merlin wins if he succeeds in convincing Arthur that the input word is in the language. Both Merlin and Arthur have access to the game history.

Definition 1.2 A language L is in AM if there is an Arthur-Merlin game with a constant number of rounds k ≥ 2 in which Arthur moves first such that

1) For $x \in L$, Merlin wins with probability at least $\frac{2}{3}$.
2) For $x \notin L$, Merlin wins with probability less than $\frac{1}{3}$.

A language L is in MA if there is a two round Arthur-Merlin game in which Merlin moves first satisfying 1) and 2) above. Note that this definition implies that $MA \subseteq AM$, since any language in MA can be decided by a three round game in which Arthur moves first and does nothing in the initial round. The reason for the asymmetry of the definition is that, if MA included all

constant-round games, it would be identical to AM, since any game in AM could be simulated by one in MA in which Merlin does nothing in the initial round.

The class AM contains important problems not known to be contained in NP, such as the problem of determining whether two graphs are isomorphic [3].

There is a simpler definition of AM and MA which is often easier to work with.

Definition 1.3 For any probability distribution D, let $\Pr_{x \in D}[A(x)]$ denote the probability of event A when x is chosen from D. Here, the uniform distribution on a set is represented by the set itself, and when this set is obvious from the context, it will be omitted.

Theorem 1.4 For $L \in \text{AM}$, there exists a polynomial-time decidable predicate M such that

$$\begin{aligned} x \in L &\leftrightarrow \Pr_{z \in \{0,1\}^m}[\exists y \in \{0,1\}^m M(x,y) = 1] \geq \frac{2}{3} \\ x \notin L &\leftrightarrow \Pr_{z \in \{0,1\}^m}[\exists y \in \{0,1\}^m M(x,y) = 1] \leq \frac{1}{3} \end{aligned} \tag{1.5}$$

where m is polynomial in $|x|$. Similarly, for $L \in \text{MA}$, there exists a polynomial time decidable predicate M such that

$$\begin{aligned} x \in L &\leftrightarrow \exists y \in \{0,1\}^m \left[\Pr_{z \in \{0,1\}^m}(M(x,y) = 1) \geq \frac{2}{3}\right] \\ x \notin L &\leftrightarrow \exists y \in \{0,1\}^m \left[\Pr_{z \in \{0,1\}^m}(M(x,y) = 1) \leq \frac{1}{3}\right] \end{aligned} \tag{1.6}$$

for m polynomial in $|x|$. The classes of languages satisfying (1.5) and (1.6) are denoted BP·NP and NP·BP, respectively. Thus, AM = BP·NP and MA = NP·BP.

Proof The theorem can be proved by reducing the problem of deciding the outcome of an Arthur-Merlin game to that of evaluating a quantified boolean formula with one block of quantifiers for each round of the game. It is then shown, using various quantifier swapping and substitution rules, that such formulas reduce to equivalent formulas with two blocks of quantifiers. For the details, see Appendix A.

1.2 Randomized Polynomial Hierarchy

The polynomial hierarchy (PH) is a hierarchy of complexity classes that generalizes NP.

Definition 1.5 The polynomial hierarchy is the set of complexity classes denoted by Σ_i, $i \geq 0$, that are defined recursively as follows.

1) $\Sigma_0 = \mathrm{P}$

2) $\Sigma_{i+1} = \mathrm{NP}^{\Sigma_i}$, $i \geq 0$

where NP^{Σ_i} denotes the class of languages decidable by a nondeterministic polynomial time Turing Machine with an oracle for Σ_i. The complement of Σ_i is denoted by Π_i.

A useful alternative characterization of PH is given by the following theorem.

Theorem 1.6 A language L is in Σ_i iff there exists a polynomial time decidable predicate M such that

$$x \in L \leftrightarrow \exists z_1 \forall z_2 \exists z_3 \cdots Q_i z_i M\left(x, z_1, z_2, ...z_i\right) \tag{1.7}$$

where z_j, $1 \leq j \leq i$, denotes a string of boolean variables and $Q_i = \exists$ ($Q_i = \forall$) for odd (even) i.

Proof See, for example, [3].

A machine model for PH can be defined based on Theorem 1.6. An alternating Turing Machine (ATM) is a generalization of a nondeterministic Turing Machine which can evaluate an expression of the form (1.7) in polynomial time. On input x, an ATM nondeterministically guesses a string of bits in a series of alternating universal and existential phases and then decides x based on the guessed bits. Those bits guessed in the universal (existential) phases correspond to the values of the universally (existentially) quantified variables in (1.7).

Remark Hierarchies analogous to PH can be defined for arbitrary time bounds $f(n)$ by replacing NP with NTIME($f(n)$) in Definition 1.5. In particular, substituting NEXP for NP in Definition 1.5 gives the exponential hierarchy (EH), for which the notation $\Sigma_i^{\mathrm{EXP}} \equiv \mathrm{NEXP}^{\Sigma_i}$ and $\Pi_i^{\mathrm{EXP}} \equiv \text{co-NEXP}^{\Sigma_i}$

is used. Such hierarchies satisfy Theorem 1.6 for a QBF of length $f(n)$, and are decidable by ATM's with time bound $f(n)$.

The expressions for AM and MA in terms of NP given in Theorem 1.4 suggest a natural generalization based on PH. This thesis is concerned mainly with the following probabilistic versions of PH.

Definition 1.7 The probabilistic polynomial hierarchy (BPH) is the hierarchy of complexity classes $\mathrm{BP}\cdot\Sigma_i$, $i \geq 1$, where $\mathrm{BP}\cdot\Sigma_i$ is the set of languages L such that there exists a predicate M decidable in Σ_i such that, for $m = \mathrm{poly}(|x|)$,

$$\begin{aligned} x &\in L \leftrightarrow \Pr_{z\in\{0,1\}^m}\left[M\left(x,z\right)\right] \geq \frac{2}{3} \\ x &\notin L \leftrightarrow \Pr_{z\in\{0,1\}^m}\left[M\left(x,z\right)\right] \leq \frac{1}{3} \end{aligned}$$

The proof of Theorem 1.4 relativizes. That is, the collapse of all classes of k-round Arthur-Merlin games to 2 rounds still holds if Arthur and Merlin are given an arbitrary oracle. This is because the proofs of Theorems A.4 to A.8 remain valid if the quantified polynomial time decidable predicate is replaced by a predicate decidable in polynomial time with an oracle. In particular, the complexity classes AM^{Σ_i} and MA^{Σ_i}, can be defined as relativizations of BP·NP and NP·BP.

Theorem 1.8 $\mathrm{BP}\cdot\Sigma_{i-1} = \mathrm{AM}^{\Sigma_i}$, $i \geq 1$.

Proof From Theorem 1.4, AM = BP·NP. Therefore,

$$\begin{aligned} \mathrm{AM}^{\Sigma_{i-1}} &= (\mathrm{BP}\cdot\mathrm{NP})^{\Sigma_{i-1}} \\ &= \mathrm{BP}\cdot\mathrm{NP}^{\Sigma_{i-1}} \\ &= \mathrm{BP}\cdot\Sigma_i \end{aligned}$$

The second line follows from the fact that a language $L \in (\mathrm{BP}\cdot\mathrm{NP})^{\Sigma_{i-1}}$ can be decided by first generating all the necessary random bits without using the oracle and then carrying out the remainder of the computation in $\mathrm{NP}^{\Sigma_{i-1}}$.

The definition of BPH given in Definition 1.7 is two-sided in the sense that there is a nonzero probability of error both for inputs in L and for inputs not in L. BPH has an equivalent one-sided definition for which there is no error for inputs in L.

Theorem 1.9 For $L \in \mathrm{BP} \cdot \Sigma_i$, there is a predicate M decidable in Σ_i such that, for $m = \mathrm{poly}(|x|)$,

$$\begin{aligned} x &\in L \leftrightarrow \forall z \in \{0,1\}^m [\mathrm{M}(x,z)] \\ x &\notin L \leftrightarrow \Pr_{z \in \{0,1\}^m} [\mathrm{M}(x,z)] \leq \frac{1}{3} \end{aligned}$$

Proof This is a trivial generalization of the proof of Theorem A.8 ii).

1.3 Derandomization

To derandomize a probabilistic complexity class is to simulate the class deterministically, without randomness. Derandomizing probabilistic complexity classes such as BPP is of considerable practical importance, since real computers do not have access to truly random bits. By Theorem 1.1, the error probability of BPP algorithms can be made exponentially small, strongly suggesting that BPP = P. Despite this, the best known upper bound for BPP is BPP$\subseteq$EXP, which follows from the brute force method of running the simulated algorithm on all possible strings of random bits.

Currently, the only known nontrivial method for derandomizing randomized complexity classes is to use a pseudorandom generator (PRG). Informally, a PRG is a function that takes as input short random strings of bits, or seeds, and outputs longer strings that appear random to any probabilistic algorithm in a given complexity class. The formal definition requires the concept of a boolean circuit.

Definition 1.10 A circuit is a collection of AND, OR, NOT, and input gates, connected by input and output lines along which bits are sent between the gates. All gates have two output lines. The input gates have no input lines and output the bits of the input to the circuit. The AND, OR, and NOT gates have one or two input lines and output the result of applying the associated boolean operation to their input(s). The output of the circuit is the output of a set of special gates, called the output gates. For any language A, an A-oracle circuit is a circuit with an additional type of gate called an oracle gate. The oracle gates have an unbounded number of input lines and output the response of the oracle to the query given by their input strings. The size of a circuit C, denoted size(C), is its number of gates.

Definition 1.11 A pseudorandom generator (PRG) is a function $G : \{0,1\}^{l(n)} \rightarrow \{0,1\}^n$, $n \geq 1$, such that, for any circuit C of size n,

$$|\Pr_{y \in \{0,1\}^n}[C(y) = 1] - \Pr_{x \in \{0,1\}^{l(n)}}[C(G(x)) = 1]\,| < \frac{1}{n} \tag{1.8}$$

A PRG allows a more efficient simulation of BPP than the brute force method, since the algorithm only needs to be simulated using outputs of the PRG, rather than all bit strings, as the random strings.

Theorem 1.12 [22] If there exists a PRG $G : \{0,1\}^{l(n)} \rightarrow \{0,1\}^n$ uniformly computable in $\mathrm{DTIME}\left(2^{(l(n))^p}\right)$ for some $p \geq 1$, then any language L decidable by a BPP algorithm in time $t(n)$ can be decided in $\mathrm{DTIME}\left(2^{O\left(l(t^2)^p\right)}\right)$.

Proof The following is a proof of Nisan and Wigderson [22]. Any time t algorithm A can be simulated by a circuit of size t^2 [23]. Therefore, taking $n = t^2$, the condition (1.8) implies that on any input, the acceptance probability of the algorithm taken over uniform random strings is within $\frac{1}{n}$ of its acceptance probability on outputs of the generator. Algorithm A can be simulated deterministically by running it on all strings in the range of G, counting the number of accepting computations, and taking the majority vote. Since there are $2^{l(t^2)}$ seeds and G is computable in time $2^{O\left(l\left(t^2\right)^p\right)}$, the simulation takes time $2^{O\left(l\left(t^2\right)^p\right)}$.

An important special case of Theorem 1.12 is $l = O(\log n)$ and $p = 1$, for which there is a complete derandomization of BPP, i.e. P = BPP.

It is not known if there exists a PRG that can "stretch" strings of bits sufficiently to allow a nontrivial derandomization of BPP. However, it has been shown that if there is a function $f \in \mathrm{DTIME}\left(2^{O(n)}\right)$ that is not computable by circuits of a certain minimum size, then such a PRG exists [22][11]. For example, if such an f is not computable by circuits of size 2^{kn} for some constant k, then there is a PRG $G : \{0,1\}^{O(\log n)} \rightarrow \{0,1\}^n$ computable in time polynomial in n, and thus P = BPP. The intuitive reason for this is that a function $G : \{0,1\}^l \rightarrow \{0,1\}^n$ can be constructed from f so that, if there is an algorithm that can distinguish outputs of G from random, then such an algorithm can be used to predict the value of f, contradicting the circuit-hardness of f.

Although the circuit lower bound condition for derandomization was a significant advance, it has so far not led to any unconditional derandomization results for standard randomized complexity classes such as BPP, MA, and AM. For a PRG to be useful for derandomization, it must be efficiently computable and its outputs must be indistinguishable from random. To satisfy these properties, it must be constructed from a function that is both uniformly computable within a certain time bound and not computable by any circuit below a certain size. The standard technique for proving the existence of such a function is to construct an algorithm that, by evaluating circuits on its inputs, ensures that, for every circuit in a given set, there is an input on which the circuit returns a different answer to that returned by the algorithm. This technique, called diagonalization, works just as well if the algorithm and the circuits are given access to an arbitrary oracle, and thus it can only be used to prove results that continue to hold if this modification is made. Because there are oracles relative to which no function exists with the uniform computability and circuit lower bound properties needed for derandomization, diagonalization is not useful in this case.

One way to circumvent the difficulty in proving derandomization results via lower bounds is to consider instead derandomization of nonstandard complexity classes such as the probabilistic hierarchies introduced in section 1.2. In these cases, the greater power of ATM's relative to deterministic and nondeterministic machines can be exploited to prove derandomization results via diagonalization. These results are important because of the relation of these hierarchies to AM and MA. For example, if it could somehow be proved that BPH collapses to AM, then derandomization results for AM would follow.

This thesis will investigate the derandomization of the hierarchies AM^{Σ_i} and MA^{Σ_i} . Chapter 2 will summarize the work of other authors in establishing the corresponence between circuit lower bounds and derandomization. The purpose of this chapter is to put the results of the thesis in context and to establish several theorems that will be needed to prove them. The main contributions of the thesis are contained in Chapter 3. There, it will be shown that a derandomization of MA^{Σ_i} is equivalent to circuit lower bounds for Σ_i^{EXP}, the exponential time version of the polynomial hierarchy. This extends an equivalence result of Impagliazzo et. al. concerning MA, and the proof technique used is similar to theirs [10]. This equivalence will be used together with a diagonalization of Kannan to derive a new unconditional derandomization of MA^{Σ_i}. This diagonalization will also be used to prove a new set of tradeoffs between derandomizations of the hierarchy AM^{Σ_i} and of

BPP. Chapter 4 will discuss possibilities for further research.

Chapter 2

Derandomization from Lower Bounds

The correspondence between circuit lower bounds for uniformly computable functions and derandomization was established in a series of papers. It was first proved that a function that cannot be approximated by a circuit below a certain size can be used to construct a PRG that derandomizes BPP [22]. A number of later papers developed this result further by showing that the existence of a function that is only hard to compute, rather than hard to approximate, is sufficient for derandomization [1][7][9][11]. In this chapter, we summarize these results, proving several hardness-randomness tradeoffs that will be used in the following chapter. These tradeoffs are given in Theorems 2.34, 2.37, 2.38, and 2.39.

2.1 The Nisan-Wigderson Theorem

The Nisan-Wigderson Theorem establishes a range of tradeoffs between deterministic simulations of BPP and hardness of approximation of uniformly computable functions f by nonuniform circuits, with greater circuit hardness of f implying a more efficient simulation. The theorem is proved by using f to construct a function $G : \{0,1\}^{l(n)} \to \{0,1\}^n$. If G is not a PRG, then its output can be distinguished from random by some BPP algorithm. This algorithm is then used to construct a nonuniform circuit which predicts the value of f with significantly higher than average probability, contradicting the hardness of approximation of f. The smaller l is in relation to n, the harder f must be, since f must "fool" a circuit that is larger relative to its input size.

The required hardness properties are defined as follows.

Definition 2.1 [22] A function $f : \{0,1\}^n \to \{0,1\}$ is (ϵ, S)-hard if for any circuit C of size S,

$$|\Pr[C(x) = f(x)] - \frac{1}{2}| < \frac{\epsilon}{2} \tag{2.1}$$

for x chosen uniformly from $\{0,1\}^n$.

Definition 2.2 [22] Let $f : \{0,1\}^* \to \{0,1\}$ be a boolean function, and let f_m be the restriction of f to inputs of length m. The hardness of f at m, $H_f(m)$, is the largest integer h_m such that f_m is $\left(\frac{1}{h_m}, h_m\right)$-hard.

The Nisan-Wigderson (NW) generator outputs a string of bits which are the outputs of f evaluated on various subsets of the input to the generator. These subsets are required to be almost disjoint to allow the output bits to be as independent as possible.

Definition 2.3 [22] Let M be a 0-1 $n \times l$ matrix. Let S_i, $1 \leq i \leq n$, be the set of column numbers in which a 1 occurs in the i^{th} row. Then M is a (k, m) design if

$$|S_i| = m \qquad \forall \quad 1 \leq i \leq n \tag{2.2}$$

$$|S_i \cap S_j| \leq k \qquad \forall \quad i \neq j \tag{2.3}$$

Definition 2.4 [22] Let M be an $n \times l$ matrix which is a (k, m) design and take $x \in \{0,1\}^l$. Let $f : \{0,1\}^m \to \{0,1\}$ be a function defined for all $m \geq 1$. Let $f_M(x)$ be the n-bit string whose i^{th} bit is obtained by applying f to the bits of x corresponding to the columns containing a 1 in the i^{th} row of M. Then the function $G : \{0,1\}^l \to \{0,1\}^n$ given by $G(x) = f_M(x)$ is called a Nisan-Wigderson (NW) generator.

The following theorem is Lemma 2.4 in Nisan and Wigderson [22].

Theorem 2.5 [22] Let $f : \{0,1\}^m \to \{0,1\}$ be a function and let A be a $n \times l$ matrix which is a $(\log n, m)$ design. Then there is a constant k such that, if $H_f(m) \geq kn^2$ for all m, the NW generator $G = f_A(x)$ is a PRG.

Proof The following proof is taken from Nisan and Wigderson [21]. Suppose for a contradiction that G is not a PRG. Then, from Definition 1.11, there is a circuit C of size n such that

$$\Pr_{x \in \{0,1\}^l}[C(G(x)) = 1] - \Pr_{y \in \{0,1\}^n}[C(y) = 1] > \frac{1}{n} \tag{2.4}$$

Let E_i be the distribution on $\{0,1\}^n$ for which the first i bits are the first i bits of $G(x)$ for x chosen uniformly at random from $\{0,1\}^l$, and the remaining bits are chosen uniformly at random. Define

$$p_i = \Pr_{z \in E_i}[C(z) = 1] \tag{2.5}$$

Since, from (2.4), $p_n - p_0 > \frac{1}{n}$, there must exist a j such that

$$p_j - p_{j-1} > \frac{1}{n^2} \tag{2.6}$$

We define a circuit, D, that predicts the j^{th} bit of the output of the generator based on the first $j-1$ bits. Let y_i denote the i^{th} bit of the generator. D takes as input the first $j-1$ bits of $G(x)$ and $n-j+1$ random bits $r_j, ...r_n$. It computes $C(y_1, ...y_{j-1}, r_j, ...r_n)$, and returns r_j if $C(y_1, ...y_{j-1}, r_j, ...r_n) = 1$ and $\bar{r}_j$, the complement of r_j, otherwise. We now prove that D has a significantly higher probability of predicting y_j correctly than a random guess. Take $z = y_1 y_2 \cdots y_{j-1} r_j \cdots r_n$. Then

$$\begin{aligned}
& \Pr_{z \in E_{j-1}}[D(z) = y_j] \\
= \; & \Pr_{z \in E_{j-1}}[C(z) = 1 \cap r_j = y_j] + \Pr_{z \in E_{j-1}}[C(z) = 0 \cap r_j = \bar{y_j}] \\
= \; & \Pr_{z \in E_{j-1}}[C(z) = 1 | r_j = y_j] \Pr(r_j = y_j) + \\
& \Pr_{z \in E_{j-1}}[C(z) = 0 | r_j = \bar{y_j}] \Pr(r_j = \bar{y_j}) \\
= \; & \frac{1}{2}\Pr_{z \in E_j}[C(z) = 1] + \frac{1}{2}\Pr_{z \in E_{j-1}}[C(z) = 0 | r_j = \bar{y_j}]
\end{aligned} \tag{2.7}$$

A lower bound for the second term in (2.7) can be obtained by noting that

$$\begin{aligned}
& \Pr_{z \in E_{j-1}}[C(z) = 1] \\
= \; & \Pr_{z \in E_{j-1}}[C(z) = 1 \cap r_j = y_j] + \Pr_{z \in E_{j-1}}[C(z) = 1 \cap r_j \neq y_j] \\
= \; & \Pr_{z \in E_{j-1}}[C(z) = 1 | r_j = y_j] \Pr(r_j = y_j) + \\
& \Pr_{z \in E_{j-1}}[C(z) = 1 | r_j = \bar{y_j}] \Pr(r_j = \bar{y_j}) \\
= \; & \frac{1}{2}\Pr_{z \in E_j}[C(z) = 1] + \frac{1}{2}\Pr_{z \in E_{j-1}}[C(z) = 1 | r_j = \bar{y_j}]
\end{aligned} \tag{2.8}$$

and substituting into (2.6):

$$\Pr_{z \in E_j}[C(z) = 1] - \frac{1}{2}\Pr_{z \in E_j}[C(z) = 1] -$$

$$\frac{1}{2}\Pr_{z\in E_{j-1}}[C(z)=1|r_j=\bar{y_j}]>\frac{1}{n^2}$$
$$\Pr_{z\in E_j}[C(z)=1]-\Pr_{z\in E_{j-1}}[C(z)=1|r_j=\bar{y_j}]>\frac{2}{n^2}$$
$$1-\Pr_{z\in E_{j-1}}[C(z)=1|r_j=\bar{y_j}]>1+\frac{2}{n^2}-\Pr_{z\in E_j}[C(z)=1]$$
$$\Pr_{z\in E_{j-1}}[C(z)=0|r_j=\bar{y_j}]>1+\frac{2}{n^2}-\Pr_{z\in E_j}[C(z)=1]$$

Substituting this into (2.7),

$$\Pr_{z\in E_{j-1}}[D(z)=y_i]\geq\frac{1}{2}+\frac{1}{n^2} \tag{2.9}$$

Clearly, there is a specific set of values of $r_j,...r_n$ for which (2.9) holds when the probability is taken over only the inputs to the generator. Fixing $r_j,...r_n$ to these values yields a deterministic circuit D' with the same minimum prediction probability.

We now transform D' to a circuit that predicts y_j based on the input to f. From Definitions 2.3 and 2.4, y_j depends on m of the generator input bits $x_1,x_2,...x_l$, which, without loss of generality, can be assumed to be $x_1,x_2,...x_m$. Thus,

$$y_j=f(x_1,x_2,...x_m) \tag{2.10}$$

As was the case for the bits $r_j,...r_n$, the bits $x_{m+1},...x_l$ can be fixed to specific bits $c_{m+1},...c_l$ while maintaining the lower bound on the prediction probability. When this is done, the resulting circuit satisfies (2.9) with the probability taken over all values of $x_1,...x_m$. A circuit D'' can now be constructed which takes as input $x_1,...x_m$ and outputs the value y_j with probability at least $\frac{1}{2}+\frac{1}{n^2}$. By Definition 2.3, the bits $y_1,...y_{j-1}$ depend only on $\log n$ of the variables $x_1,...x_m$. Since any boolean function on k bits can be computed by a circuit of size 2^k [6], each of the bits $y_1,...y_{j-1}$ can be computed by circuits of size n. The circuit D'' is obtained from D' by replacing the gates for $y_1,...y_{j-1}$ by these circuits, and introducing new input gates $x_1,...x_m$ which provide inputs to the circuits. Since $j\leq n$, the size of D'' is less than tn^2 for some constant t. Together with the prediction probability, this contradicts the hardness assumption for f for $k=t$.

The next two theorems show that the design required by Theorem 2.5 exists and can be efficiently generated.

Theorem 2.6 (Lemma 2.5 [22]) For all integers n and m such that $\log n \leq m \leq n$, there exists an $n \times l$ matrix which is a $(\log n, m)$ design with $l = O(m^2)$, and this design is constructible in time polynomial in n.

Theorem 2.7 (Lemma 2.6 [22]) For all integers n and m, where $m = C \log n$ for some constant C, there exists a $n \times l$ matrix which is a $(\log n, m)$ design with $l = O\left(C^2 \log n\right)$, and this design is constructible in time polynomial in n.

Theorems 2.5, 2.6, and 2.7 imply a range of tradeoffs between circuit hardness of uniformly computable functions and derandomizatons of BPP.

Theorem 2.8 (Theorem 2 [22]) If there exists a function f such that, for all but finitely many m,

1) $f \in$ EXP and $H_f(m) \geq m^c$ for any fixed c,
2) $f \in$ EXP and $H_f(m) \geq 2^{m^\epsilon}$ for some $\epsilon > 0$, or
3) $f \in \mathrm{E} \equiv \mathrm{DTIME}\left(2^{O(n)}\right)$ and $H_f(m) \geq 2^{\epsilon m}$ for some $\epsilon > 0$, respectively

then respectively

1) $\mathrm{BPP} \subseteq \bigcap_{\epsilon > 0} \mathrm{DTIME}\left(2^{n^\epsilon}\right)$
2) $\mathrm{BPP} \subseteq \mathrm{DTIME}\left(2^{(\log n)^c}\right)$ for some constant c.
3) BPP = P

Proof 1) Let f be a function satisfying hypothesis 1 of the theorem. By Theorems 2.5 and 2.6, there is a PRG $G : l \to n$ with

$$\begin{aligned} n^2 = O(m^c) &= O(l^{\frac{c}{2}}) \\ n &= O(l^{\frac{c}{4}}) \\ l &= kn^{\frac{4}{c}}, \text{ for some constant k} \end{aligned}$$

From Theorem 1.12, any $t(n)$ time BPP algorithm can be simulated in $\mathrm{DTIME}\left(2^{O\left(t^{\frac{8p}{c}}\right)}\right)$ for some $p \geq 1$. For any $\epsilon > 0$, c can be made sufficiently large to obtain a $\mathrm{DTIME}\left(2^{n^\epsilon}\right)$ simulation.

2) The proof is similar to that of 1).

3) The proof is similar to that of 1) and 2) except that Theorem 2.7 is used instead of Theorem 2.6.

A milder derandomization follows if it is assumed that the function f is hard only for infinitely many, rather than for all but finitely many, input lengths. For a complexity class C, let io-C denote the set of languages L such that there exists a language $L' \in$C with $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$ for infinitely many n.

Theorem 2.9 If there exists a function f such that, for infinitely many m,

1) $f \in$ EXP and $H_f(m) \geq m^c$ for any fixed c
2) $f \in$ EXP and $H_f(m) \geq 2^{m^\epsilon}$ for some $\epsilon > 0$, or
3) $f \in$ E and $H_f(m) \geq 2^{\epsilon m}$ for some $\epsilon > 0$, respectively

then respectively

1) BPP $\subseteq \bigcap_{\epsilon>0}$ io-DTIME $\left(2^{n^\epsilon}\right)$
2) BPP $\subseteq$ io $-$ DTIME $\left(2^{(\log n)^c}\right)$ for some constant c.
3) BPP = io-P

Proof The proof of Theorem 2.8 applies here for all input lengths for which f satisfies the hardness condition.

2.2 Hardness Amplification

In the last section, it was proved that hardness of approximation, or average-case hardness, of a function in E is sufficient to construct a PRG and derandomize BPP. In work done by a number of different authors, this result was strengthened to show that all that is required is that this function be hard to compute, or worst-case hard [1][7][9][11]. The proof uses various "hardness amplification" theorems which show that a function $g \in$ E that possesses the required average case hardness can be constructed from a worst-case hard function $f \in$ E. The main idea used to construct g is that it is harder to solve several instances of a problem than to solve a single instance, so that a function that incorporates the value of a function on many independent inputs is harder to compute than the original function. The hardness

amplification is carried out in a series of successive phases, each using a different technique to increase the hardness achieved in the previous phase. In this section, we present this hardness amplification proof, thus showing that Theorem 2.8 holds with hardness of approximation replaced by hardness of computation. For simplicity, we focus on the "high end" derandomization condition Theorem 2.8 3), although the proof can be easily generalized to show the same result for Theorem 2.8 1) and 2) as well.

2.2.1 Random Self-Reduction

The first step in hardness amplification involves showing that any E-complete function that is hard to compute also possesses a mild average case hardness.

Definition 2.10 Let $f : \{0,1\}^* \to \{0,1\}$ be a function. The restriction of f to inputs of length n is denoted by f_n.

Definition 2.11 The term "almost everywhere" (a.e.) will be used to mean for all but finitely many n. The term "infinitely often" (i.o.) will be used to mean for infinitely many n.

The following notation is from Impagliazzo and Wigderson [11].
Definition 2.12 Let $f : \{0,1\}^m \to \{0,1\}^n$ be a function. The worst case circuit complexity of f, denoted $\mathrm{S}(f)$, is the minimum size of a circuit computing f. The success of f for size s, $\mathrm{SUC}_s(f)$, is the maximum over all circuits C of size s of $\Pr[C(x) = f(x)]$. In the case $n = 1$, the advantage of f for size s, $\mathrm{ADV}_s(f)$, is defined as $\Pr[C(x) = f(x)] - \Pr[C(x) \neq f(x)] = [2\mathrm{SUC}_s(f) - 1]$. For a function $f : \{0,1\}^* \to \{0,1\}$, expressions involving $\mathrm{S}(f)$, $\mathrm{SUC}_s(f)$, and $\mathrm{ADV}_s(f)$ hold for $\mathrm{S}(f_n)$, $\mathrm{SUC}_s(f_n)$, and $\mathrm{ADV}_s(f_n)$ a.e.

Theorem 2.13 [1] Suppose there is an E-complete function f with $S(f) \in 2^{\Omega(n)}$. Then for some $s' \in 2^{\Omega(n)}$,

$$\mathrm{SUC}_{s'}(f) < 1 - \frac{1}{3n} \tag{2.11}$$

The proof of Theorem 2.13 uses a property of E-complete functions called random self-reducibility. This is the property that, if there is a circuit that computes such a function correctly on most inputs, this circuit can be used to construct a random circuit that computes the function correctly with high probability on all inputs. A random circuit is a circuit that inputs a string

of random bits. That E-complete functions are random self-reducible follows from the random self-reducibility of the multilinear extensions over finite fields of their characteristic functions, to which they are Turing equivalent [1].

For a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, a multilinear extension of f over a field F is a function $g : F^n \rightarrow \{0,1\}$ over F that is linear in its inputs $x_1, x_2, ...x_n$ and agrees with f on $\{0,1\}^n$. Every boolean function on n variables has a unique multilinear extension over any finite field [1]. The multilinear extension of a function $f : \{0,1\}^* \rightarrow \{0,1\}$ is the function $g : F^* \rightarrow F$ that computes on input of length n the multilinear extension of f_n.

Theorem 2.14 [1] Let f be a function deciding an E-complete language L and let g be its multilinear extension over a finite field. Then f and g are polynomial time Turing reducible to each other.

Proof Obviously, $f \in \mathrm{P}^g$. We give a polynomial time alternating Turing machine (ATM) for deciding g with an oracle for f and then show that this machine can be simulated in P^L. The following algorithm is from Babai et. al. [1]. On input $x_1, x_2, ...x_n$, the machine first existentially guessses the value $g(x_1, ...x_n)$ and the linear function $h_1(y) = g_n(y, x_2, ...x_n)$, and checks that $h_1(x_1) = g_n(x_1, ...x_n)$. It then universally guesses $t_1 \in \{0,1\}$, existentially guesses $h_2(y) = g(t_1, y, x_3, ...x_n)$, and checks that $h_2(x_2) = h_1(t_1)$. This process continues for $n-1$ steps where, in the i^{th} step, the machine universally guesses a $t_i \in \{0,1\}$, existentially guesses $h_{i+1}(y) = g(t_1, t_2, ...t_i, y, x_{i+2}, ...x_n)$, and checks that $h_{i+1}(x_{i+1}) = h_i(t_i)$. By the end of the process, the machine will have verified that the guessed linear functions h_i, $1 \leq i \leq n$, define a multilinear function g_n on Z_p. In the final step, the machine verifies that g_n is the multilinear extension of f_n by querying the oracle for f on $t_1 t_2 \cdots t_n$ and checking that the answer agrees with $h_n(t_1, t_2, ...t_n) = g_n(t_1, t_2, ...t_n)$.

Since $L \in$ E, the oracle ATM can be simulated in EXP, and $g \in \mathrm{P}^f$ now follows from the fact that EXP$\subseteq \mathrm{P}^f$. To see that EXP$\subseteq \mathrm{P}^f$, let $A \in$EXP be a language decidable in time $2^{p(n)}$ for some polynomial $p(n)$. The set A' consisting of strings $x \in$A padded to length $p(|x|)$ is decidable in E. By the E-completeness of L, there is a polynomial time many-one reduction R from A' to L. To decide A, a polynomial-time L-oracle machine pads its input x to length $p(|x|)$, applies R to the padded string, queries the oracle on the

resulting string, and returns the answer.

The following two theorems establish the random self-reducibility of multilinear extensions of boolean functions over sufficiently large finite fields.

Theorem 2.15 [18] Let $f(x_1, ...x_n)$ be a polynomial of degree d over a finite field of size $q > d+1$, and let $a_1, ...a_{d+1}$ be any distinct nonzero elements of the field. Then there are weights $w_1, w_2, ...w_{d+1}$ such that

$$f(x_1, ...x_n) = \Sigma_{i=1}^{d+1} w_i f(\tilde{x}_i) \tag{2.12}$$

where $\tilde{x}^i = (x_1 + a_i\alpha_1, ...x_n + a_i\alpha_n)$ for arbitrary field elements $\alpha_1, ...\alpha_n$.

Theorem 2.16 [1] Let g_n be the multilinear extension of a boolean function on n variables over the field Z_p for p a prime greater than $n+1$. Suppose there is a circuit C computing g_n correctly for all but a fraction $\frac{1}{3n}$ of inputs. Then for some fixed polynomial $p(n)$, there is a random circuit of size $p(n) \times \text{size}(C)$ that computes g_n correctly with probability greater than $1 - 2^{-n}$.

Proof The multilinear extension over a field of a boolean function on n inputs has degree n [1]. Therefore, Theorem 2.15 can be applied to g_n to obtain a random circuit for g_n that takes the $\alpha_i's$ as random inputs and uses C to evaluate $g_n(\tilde{x}_i)$. This circuit computes g_n correctly with probability at least $1 - \frac{n+1}{3n} \geq \frac{3}{5}$ taken over the random inputs. By the same argument as was used to prove Theorem 1.1, the success probability can be amplified to over $1 - 2^{-n}$. The bound on the size of the resulting circuit follows from the fact that computing the right-hand side of (2.12) requires a circuit of size a polynomial times size(C), and amplifying the success probability multiplies this by a polynomial factor.

Proof (of Theorem 2.13) We prove the contrapositive. Suppose there is an E-complete function f such that, for any $\epsilon > 0$ and for infinitely many n, there is a circuit of size $2^{\epsilon n}$ satisfying

$$Pr[C(x) = f(x)] \geq 1 - \frac{1}{3n} \tag{2.13}$$

By the proof of Theorem 2.14, the multilinear extension g_n of f_n over Z_p for $p > n+1$ is decidable in polynomial time with a single query to an oracle for f. For any $\epsilon' > 0$, a $2^{\epsilon' n}$ size circuit that computes g_n correctly on all but a $\frac{1}{3n}$

fraction of inputs can be constructed using a circuit for f_n satisfying (2.13). By Theorem 2.16, there is a random circuit C' of size $p(x)2^{\epsilon' n}$ that computes g_n with probability greater than $1 - 2^{-n}$. The probability that C' fails to compute g_n on some input is less than $2^n \cdot 2^{-n} = 1$. Therefore, there must be some random bit string for which C' succeeds on all inputs. Hardwiring this bit string into C' gives a circuit that computes g and therefore f. Since ϵ' can be made arbitrarily small, $S(f) \notin 2^{\Omega(n)}$.

2.2.2 Hard-core Sets

The next step in hardness amplification enhances the hardness achieved in the previous section to a constant error.

Theorem 2.17 [9] Let $f \in \mathrm{E}$ be a function with $\mathrm{SUC}_s(f_n) < 1 - \frac{1}{16n}$ for some $s \in 2^{\Omega(n)}$. Then there is a function $g \in \mathrm{E}$ such that $\mathrm{SUC}_{s'(n)}(g_n) < 1 - \frac{0.05}{16}$ for some $s' \in 2^{\Omega(n)}$.

To achieve this amplification, f is first shown to possess a "hard-core set" on which it is more difficult to compute than average.

Theorem 2.18 [9] Let $f : \{0,1\}^n \to \{0,1\}$ be a function such that $\mathrm{SUC}_s(f) \leq 1 - \delta$. For any $\epsilon > 0$, there is a set $S \subseteq \{0,1\}^n$, called a hard-core set, with $|S| \geq \delta 2^n$ such that $\mathrm{SUC}_{d\epsilon^2\delta^2 s}(f) < \frac{1}{2} + \frac{1}{2}\epsilon$, where the success probability is taken over all $x \in S$ and d is a constant.

The function g in Theorem 2.17 is a function of f evaluated on many different pairwise independent inputs. With significant probability, one of the inputs on which f is evaluated will be in a hard-core set of f. Roughly speaking, this implies that g must be hard to compute, since otherwise the value of f on its hard-core set could be easily derived from the output of g, contradicting the hardness of f on this set.

The required pairwise independence is achieved using the following theorem.

Theorem 2.19 [9] There exists a polynomial time computable function $p : \{0,1\}^{O(n)} \to (\{0,1\}^n)^n$ that outputs a set of n bit strings $x_1, x_2, ...x_n$ that are pairwise independent. That is, for $i \neq j$, $\Pr[x_i = a \cap x_j = b] = \Pr[x_i = a] \cdot \Pr[x_j = b]$, where the probability is over all $O(n)$ bit inputs to p. Moreover,

there is a polynomial time algorithm q that inputs an index i and a value c for x_i and outputs a value of r such that $x_i = c$ in $p(r)$.

The hard function will be constructed from the inner product mod 2 of two vectors $r, s \in \{0,1\}^n$, denoted $\langle r, s \rangle$. The following theorems show how to compute a bit vector with high probability from its inner product with a random vector.

Theorem 2.20 [7] Let $v \in \{0,1\}^n$ and let $B : \{0,1\}^n \to \{0,1\}$ be a function such that $\Pr_{s \in 0,1^n}[B(s) = \langle s, v \rangle] \geq \frac{1}{2} + \gamma$. Then there exists a poly(n) time probabilistic Turing Machine M that, given n as input and B as oracle, outputs v with probability at least $O(\gamma^2)$.

Theorem 2.21 [9] Let $v \in \{0,1\}^n$ and let $B : \{0,1\}^n \to \{0,1\}$ be a function such that $\Pr_{s \in 0,1^n}[B(s) = \langle s, v \rangle] \geq 0.8$. Then there exists a poly(n) time probabilistic Turing Machine N that, given n as input and B as oracle, outputs v with probability at least $O(\gamma^2)$.

Proof By running the machine from Theorem 2.20 $p(n)$ times for some polynomial p, a polynomial size list of strings of length n can be constructed that contains v with probability $1 - (1 - \gamma^2)^{p(n)}$. For each $w \neq v$ in this list,

$$\begin{aligned} & \Pr[B(s) = \langle s, w \rangle] \\ \leq\ & \Pr[\langle s, w \rangle = \langle s, v \rangle] + \Pr[B(s) \neq \langle s, v \rangle] \\ \leq\ & 0.5 + 0.2 \\ =\ & 0.7 \end{aligned}$$

To find v, N uses the oracle for B to find $B(s)$ for $q(n)$ values of s for some polynomial q, and compares the results to $\langle s, w \rangle$ for each w in the list, recording the number of matches. N then returns a w such that the number of matches is at least $0.75q(n)$ if there is such a w, otherwise it returns an arbitrary w. From Chernoff Bounds [21], N can be made to output v with probability greater than $1 - 2^{-n}$.

Theorem 2.22 [9] Let $h : \{0,1\}^{O(n)} \to \{0,1\}^n$ be a function. If there is a circuit C of size g such that $\Pr_{r,s}[C(r, s) = \langle s, h(r) \rangle] \geq 1 - \gamma$, then there is a circuit C' of size $n^{O(1)}g$ such that $\Pr_r[C'(r) = h(r)] \geq 1 - 5\gamma$.

Proof Let S be the set of r's on which C outputs $\langle s, h(r)\rangle$ with probability less than 0.8, and let t be the fraction of r's in S. Then,

$$\begin{aligned} 0.8t + (1-t) &> \Pr\left[C(s,r) = \langle s, h(r)\rangle\right] \\ &\geq 1-\gamma \\ -0.2t &> -\gamma \\ t &< 5\gamma \end{aligned}$$

From Theorem 2.21, for any $r \notin S$, there is a probabilistic Turing Machine N that, given n as input and $C(r,s)$ as oracle, computes $h(r)$ with probability greater than $1-2^{-n}$. Arguing as in the proof of Theorem 2.13, there is a string of random bits for which N computes $h(r)$ correctly for all $r \notin S$. The required circuit takes r as input and simulates N on n with oracle $C(r,s)$ using this string, which is hardwired into the circuit.

Theorem 2.17 follows immediately from the next theorem.

Theorem 2.23 [9] Let $f \in E$ be a function with $SUC_s(f_n) < 1-\delta$ for $\delta = \frac{1}{16n}$, where $s \in 2^{\Omega(n)}$. Let $p : \{0,1\}^{O(n)} \rightarrow (\{0,1\}^n)^n$ be a function as in Theorem 2.19. Let $g(r,t) = \langle t, f(x_1)f(x_2)\cdots f(x_n)\rangle$, where $p(r) = (x_1, x_2, ...x_n)$. Then $\mathrm{SUC}_{s'}(g) < 1 - 0.05\delta n$ for some $s' \in 2^{\Omega(n)}$.

Proof By Theorem 2.22, the theorem can be proved by showing that, for $\bar{f}(r) = f(x_1)f(x_2)\cdots f(x_n)$, $\mathrm{SUC}_{s_1}\left(\bar{f}\right) < 1 - 0.25\delta n$ for some $s_1 \in 2^{\Omega(n)}$. Applying Theorem 2.18 with $\epsilon = 0.2$, there is a set H of size $\delta 2^n$ such that, for some $s_2 \in 2^{\Omega(n)}$, $\mathrm{SUC}_{s_2}(f) < 0.6$ on H. Suppose for a contradiction that for any $s \in 2^{\Omega(n)}$, there is a circuit C of size s that computes $\bar{f}$ on all but a $0.25\delta n$ fraction of inputs. For any i, the probability that $x_i \in H$ and for all $j \neq i$, $x_j \notin H$ is given by

$$\begin{aligned} & \Pr\left(x_i \in H \cap_{j\neq i} x_j \notin H\right) \\ = \; & \Pr\left(x_i \in H\right) \Pr\left(\cap_{j\neq i} x_j \notin H | x_i \in H\right) \\ \geq \; & \Pr\left(x_i \in H\right) \left[1 - \sum_{j\neq i} \Pr\left(x_j \in H | x_i \in H\right)\right] \\ = \; & \Pr\left(x_i \in H\right) \left[1 - \sum_{j\neq i} \Pr\left(x_j \in H\right)\right] \\ = \; & \delta\left(1 - n\delta\right) \\ \geq \; & \frac{15}{16}\delta \end{aligned} \tag{2.14}$$

where the third last line follows from pairwise independence. Assume that there is no i such that, conditioned on this event occurring for i, the probability that C computes $\bar{f}$ is at least $\frac{2}{3}$. Since these events are mutually exclusive, this gives a probability that C fails to compute $\bar{f}$ of at least $\left(\frac{1}{3}\right)\left(\frac{15}{16}\right)\delta n > 0.25\delta n$, a contradiction. Therefore, there must be an i such that, conditioned on x_i being the only element in H, the probability that C fails to compute $\bar{f}$ is at most $\frac{1}{3}$. From (2.14),

$$\begin{aligned} \Pr\left(\cap_{j\neq i} x_j \notin H | x_i \in H\right) &= \frac{\Pr\left(\cap_{j\neq i} x_j \notin H \cap x_i \in H\right)}{\Pr\left(x_i \in H\right)} \\ &\geq \frac{\frac{15}{16}\delta}{\delta} \\ &= \frac{15}{16} \end{aligned} \tag{2.15}$$

Consequently,

$$\Pr\left[C(r) \neq f(r) | x_i \in H\right] \leq \frac{1}{3} + \frac{1}{16} \leq 0.4 \tag{2.16}$$

A circuit C' for $f(x_i)$ can be constructed as follows. On input a, C' uses the function q defined in Theorem 2.19 to obtain a value for r such that $x_i = a$ in $p(r)$. It then computes C on this value and outputs the i^{th} bit of the output of C. Since C can be made size s for any $s \in 2^{\Omega(n)}$, so can C', and by (2.16), the success probability of C on H is at least 0.6. This contradicts the fact that $\mathrm{SUC}_s(f) < 0.6$ on H for some $s \in 2^{\Omega(n)}$.

2.2.3 The XOR Lemma

The final phase of hardness amplification increases the constant error obtained in the last section to the exponentially small advantage required by Theorem 2.8 3). This was done by Impagliazzo and Wigderson using a technique based on the XOR Lemma, which states that the XOR of several independent instances of a decision problem is harder to compute than a single instance of the problem [11].

Theorem 2.24 (The XOR Lemma) [24][19] For $g : \{0,1\}^n \to \{0,1\}$, define the direct product function $g^{\oplus k} : \left(\{0,1\}^n\right)^k \to \{0,1\}$ by $g^{\oplus k}\left(x_1, x_2, ...x_k\right) = g(x_1) \oplus g(x_2) \oplus \cdots g(x_k)$. Then for any $g : \{0,1\}^n \to \{0,1\}$ with $\mathrm{SUC}_s(g) \leq 1 - \delta$, we have $\mathrm{ADV}_{s'}\left(g^{\oplus k}\right) \leq \epsilon$ for $k = O\left(-\frac{\log \epsilon}{\delta}\right)$, $s' = s\left(\epsilon\delta\right)^{O(1)}$.

By applying the XOR Lemma, Theorem 2.17 can be improved to obtain any constant advantage less than $\frac{1}{2}$ by XORing a constant number of instances of the function g. In particular, taking $\delta = \frac{0.05}{16}$ and $\epsilon = \frac{1}{3}$ in Theorem 2.24 and using Theorems 2.13 and 2.17 gives the following Theorem.

Theorem 2.25 If there is an $f \in E$ with $S(f) \in 2^{\Omega(n)}$, then there is a function $g \in E$ with $\mathrm{SUC}_s(g) \leq \frac{2}{3}$ for some $s \in 2^{\Omega(n)}$.

The XOR Lemma cannot be used directly to construct a function with exponentially small advantage, since, taking $\epsilon = 2^{-cn}$ and constant δ in Theorem 2.24 leads to a quadratic increase in the input size of the resulting $g^{\oplus k}$ relative to g. Impagliazzo and Wigderson deal with this problem by derandomizing the XOR Lemma [11]. It is shown that the inputs x_i in Theorem 2.24 need not be completely independent, but can be generated from a linear number of bits by a function whose outputs have certain properties. A theorem analogous to Theorem 2.24 is proved with random independent inputs replaced by the output of a direct product generator, defined as follows.

Definition 2.26 (Definition 3 [11]) For any boolean function $g : \{0,1\}^n \to \{0,1\}$, define $g^{(k)} : \left(\{0,1\}^n\right)^k \to \{0,1\}^k$ as $g^{(k)}(x_1, ...x_k) = (g(x_1), ...g(x_k))$. Let $G : \{0,1\}^m \to \left(\{0,1\}^n\right)^k$ be a function. G is an $(s, s', \epsilon, \delta)$ direct product generator if for every such function g with $\mathrm{SUC}_s(g) < 1 - \delta$, we have $\mathrm{SUC}_{s'}\left(g^k \circ G\right) < \epsilon$.

The next theorem shows how a direct product generator can be used to derandomize the XOR Lemma.

Theorem 2.27 [7] For $f : \{0,1\}^m \to \{0,1\}^l$, if $\mathrm{SUC}_s(f) \leq \delta$ then for some $s' = \epsilon^{O(1)}s$ and $\delta = \epsilon^{O(1)}$, $\mathrm{ADV}_{s'}\left(f^{\oplus}\right) \leq \epsilon$, where $f^{\oplus} : \{0,1\}^m \to \{0,1\}$ outputs the XOR of the l bits output by f.

Taking $f = g^k \circ G$ in Theorem 2.27 gives the following corollary.

Corollary 2.28 Let $G : \{0,1\}^m \to \left(\{0,1\}^n\right)^k$ be a $(s, s', \epsilon, \delta)$ direct product generator. Then for every boolean function g with $\mathrm{SUC}_s(g) \leq 1 - \delta$, there is a boolean function f with $\mathrm{ADV}_{s''}(f) \leq \epsilon^{O(1)}$, where $s'' = \epsilon^{O(1)}s'$.

From Theorem 2.25 and Corollary 2.28, the hardness amplification problem reduces to finding an efficiently computable $(s, s', \epsilon, \delta)$ direct product generator with $s = 2^{\Omega(n)}$, $s' = 2^{\Omega(n)}$, $\epsilon = 2^{-\Omega(n)}$ and $\delta = \frac{1}{3}$ that inputs $O(n)$ bits. Impagliazzo and Wigderson construct such a generator [11].

According to Definition 2.26, a direct product generator must have the property that a lower bound on the success probability of computing $g^k \circ G$ implies a lower bound on the success probability of computing g. Thus, a function G can be proven to be a direct product generator by showing a way to compute with high probability the value of a function g from the output of $g^{\oplus k}$ on inputs from G. This can be done provided G satisfies the following three properties.

Definition 2.30 (Definition 4 [11]) For any polynomial time computable function $G : \{0,1\}^m \to (\{0,1\}^n)^k$, G is M-restrictable if there is a polynomial time computable function $h(i, x, \alpha) : [n] \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$, where $[n]$ denotes the set $\{1, 2, ...n\}$, satisfying 1), 2), and 3) below. Let $(x_1, x_2, ...x_k)$ denote the output of G on input $h(i, x, \alpha)$.

1) For any given i, $h(i, x, \alpha)$ is uniformly distributed in $\{0,1\}^m$ when X and α are chosen uniformly at random.

2) $x_i = x$.

3) There is a constant M such that, for any given i, $j \neq i$, and α, there is a set $S \subseteq \{0,1\}^n$ with $|S| \leq M$ such that $x_j \in S$ for any x.

Definition 2.31 (Definition 5 [11]) A function $G : \{0,1\}^m \to (\{0,1\}^n)^k$ producing output $(x_1, ...x_k)$ is (k', q, δ)-hitting if for any sets $H_1, ...H_k \subseteq \{0,1\}^n$ such that $|H_i| \geq \delta 2^n$, we have $\Pr[|\{i|x_i \in H_i\}| < k'] \leq q$.

Definition 2.32 A function $G : \{0,1\}^m \to (\{0,1\}^n)^k$ producing output $(x_1, ...x_k)$ is uniform if for every $i \in [n]$ the number of inputs $r \in \{0,1\}^m$ such that $x_i = a$ is the same for all $a \in \{0,1\}^n$.

The following theorem implies that a function with the above properties is a direct product generator.

Theorem 2.33 (Theorem 15 [11]) Let $G(r) : \{0,1\}^m \to (\{0,1\}^n)^k$ be a uniform, $(\rho k, q, \delta)$-hitting, M-restrictable function, where $q > 2^{\frac{-\rho k}{3}}$. Take $s > 2Mnk$. Then G is a $(s, s', \epsilon, \delta)$ direct product generator, where $\epsilon = \left[4\left(\frac{\delta}{\rho}\right) + 1\right] q$ and $s' = \frac{q^2}{n} s - kMn$.

Proof Suppose there exists a circuit C of size s' that computes $g^{(k)} \circ G$ with success probability ϵ. We construct a family F of probabilistic circuits that computes g with advantage at least q on most inputs. Denote the output

of G by $(x_1, x_2, ...x_k)$. Each circuit $C' \in F$ has hardwired into it values I, α_0, and $g(x_i)$, for all $i \neq I$, for all values of x_i that can be output by $G(h(I, x, \alpha_0))$ as x ranges over $\{0,1\}^n$. By Definition 2.30, there are at most M such values. On input x, C' evaluates $g^{(k)}$ on input $G(h(I, x, \alpha_0))$ using C. It then compares the i^{th} bit output by C for $i \neq I$ to the value of $g(x_i)$ and counts the number t of mismatches. C' outputs the I^{th} bit of C with probability 2^{-t}, and otherwise outputs a random bit.

For any set $H \subseteq \{0,1\}^n$ with $|H| \geq \delta 2^n$, the advantage for computing g on H taken over all circuits in F, inputs to the circuits, and random strings used by the circuits is at least $p - q - 2^{\frac{-\rho k}{3}}$, where p is the success probability of C conditioned on $x_I \in H$ for some I. To see this, let $b_1 b_2 \cdots b_k$ be the output of C on input r such that $G(r) = (x_1, x_2, ...x_k)$. Let T be the number of incorrect bits output by C on this input, and let T' be the number of incorrect bits b_i for i such that $x_i \in H$. Let k' be the number of x_i in H, and suppose $k' \geq \rho k$. We consider separately the two cases of b_I correct and b_I incorrect. By property 1) of Definition 2.30, I is independent of $x_1, x_2, ...x_k$, so with probability $\frac{T'}{k'}$, the I^{th} bit is incorrect and $t = T - 1$. In this case, C' fails by choosing to output the I^{th} bit of C with probability $2^{-t} = 2^{-T+1}$. Otherwise, C' outputs a random bit and has zero advantage. With probability $1 - \frac{T'}{k'}$ the I^{th} bit is correct and $t = T$. In this case, C' succeeds by choosing to output the I^{th} bit of C with probability $2^{-t} = 2^{-T}$. Otherwise, C' outputs a random bit and has zero advantage. Therefore, the overall advantage is $2^{-T}\left[\left(1 - \frac{T'}{k'}\right) - \frac{2T'}{k'}\right] = 2^{-T}\left(1 - \frac{3T'}{k'}\right)$. If $T' \leq \frac{k'}{3}$, this is non-negative. If $T' > \frac{k'}{3}$, then it is at least $-2^{-\frac{\rho k}{3}}$.

With probability p, all bits output by C are correct, in which case C' outputs the I^{th} bit of C and succeeds with probability 1. In all other cases, the advantage of C' is at least $-2^{\frac{-\rho k}{3}}$ unless $k' < \rho k$ which, by Definition 2.31, happens with probability at most q. Therefore, the overall advantage is at least $p - q - 2^{-\frac{\rho k}{3}}$.

To derive a lower bound for p, let D be the distribution $G(r)$ for random r, and let D' be this distribution conditioned on $x_I \in H$ for some I. For any $(x_1, x_2, ...x_k)$, let u be the number of x_i in H. The probability of such a sequence in D' is $\frac{u}{\delta k}$ times its probability in D. To see this, note that the probability of the sequence in D' is proportional to u times its probability in D, since there are u possibilities for I. By the uniformity of G, the expected value of u in D is δk, so the constant of proportionality must be $\frac{1}{\delta k}$ for D' to be a normalized probability distribution. The probability in D that $u \geq \rho k$

and C is correct is at least $\epsilon - q$ from the hitting property of G. Therefore, the probability in D' that $u \geq \rho k$ and C is correct is at least $\frac{(\epsilon-q)\rho}{\delta} = 4q$, which is the desired lower bound for p. This gives an advantage for computing g of at least $4q - q - 2^{-\frac{\rho k}{3}} = 3q - 2^{-\frac{\rho k}{3}} \geq q$.

Thus, the success probability for computing g taken over all circuits of F, inputs, and random choices is at least $\frac{1}{2} + \frac{q}{2}$ for all sets H with $|H| \geq \delta 2^n$. This implies that the set S of all inputs x such that the success probability taken over all circuits of F and random choices is less than $\frac{1}{2} + \frac{q}{2}$ is of size less than $\delta 2^n$. Indeed, if this were not true, then a size $\delta 2^n$ subset of S would be a set that violates the above lower bound on advantage. Define a random circuit E that, on input x, chooses at random $\frac{n}{q^2}$ circuits from F and their associated random bits, runs these circuits on x, and outputs the majority. By the proof of Theorem 1.1, this circuit decides all inputs x with success probability greater than $1 - 2^{-n}$, and so as was done in the proof of Theorem 2.13, a specific set of random bits can be hardwired into E to obtain a deterministic circuit that decides g for all $x \in \{0,1\}^n \backslash S$. This is a circuit of size $(|C| + kMn)\frac{n}{q^2} = (s' + kMn)\frac{n}{q^2} = s$ that computes g on all but a δ fraction of inputs, so the theorem is proved.

Impagliazzo and Wigderson construct a function $G : \{0,1\}^{cn} \rightarrow (\{0,1\}^n)^n$ that is a $(2^{\Omega(n)}, 2^{\Omega(n)}, 2^{-\Omega(n)}, \frac{1}{3})$, where c is a constant [11]. This completes the proof of hardness amplification of functions $f \in E$ with exponential size circuit lower bounds. The same methods can be applied to functions with subexponential lower bounds, leading to the following stengthened versions of the hardness-randomness tradeoffs of Theorem 2.8.

Theorem 2.34 If there exists a function f such that, for all but finitely many n,

1) $f \in$ EXP and $S(f_n) \geq n^k$ for fixed k

2) $f \in$ EXP and $S(f_n) \geq 2^{n^\epsilon}$ for some $\epsilon > 0$, or

3) $f \in$ E and $S(f_n) \geq 2^{\epsilon n}$ for some $\epsilon > 0$, respectively

then respectively,

1) BPP $\subseteq \cap_{\epsilon>0}$DTIME $\left(2^{n^\epsilon}\right)$

2) BPP $\subseteq$ DTIME $\left(2^{(\log n)^c}\right)$ for some constant c.

3) BPP = P

An io form of Theorem 2.34 analogous to Theorem 2.9 can also be given.

The Nisan-Wigderson and hardness amplification theorems relativize [17]. That is, for any oracle A, an NW-generator that is constructed from an A-oracle circuit-hard function f produces outputs that cannot be distinguished from random by A-oracle circuits.

Definition 2.35 A function $G : \{0,1\}^{l(n)} \to \{0,1\}^n$ is an A-PRG if for any A-oracle circuit C of size n with n inputs,

$$|\Pr_{x\in\{0,1\}^{l(n)}}[C(G(x)) = 1] - \Pr_{y\in\{0,1\}^n}[C(y) = 1]\,| < \frac{1}{n} \tag{2.17}$$

Definition 2.36 For a boolean function f, $H_f^A(n)$ and $S^A(f)$ are defined as in Definitions 2.2 and 2.12 with circuits replaced by A-oracle circuits.

Theorem 2.37 The hardness-randomness tradeoffs of sections 2.1 and 2.2 relativize. In particular, if there exists f such that

1) $f \in$ EXP and $S^A(f) \geq n^k$ for any fixed k

2) $f \in$ E and $S^A(f) \geq 2^{\epsilon n}$ for some $\epsilon > 0$

then there exists an A-PRG

1) $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ computable in DTIME $\left(2^{n^\epsilon}\right)$ for every $\epsilon > 0$

2) $G : \{0,1\}^{O(\log n)} \to \{0,1\}^n$ computable in deterministic polynomial time.

It is sometimes useful to consider constructions of PRG's from functions that are not necessarily deterministically computable or even uniformly computable. The following theorems deal with this case.

Theorem 2.38 (Theorem 11 [10]) Let A be an oracle. For every $\epsilon > 0$, there is a polynomial time computable function $F : \{0,1\}^{2^{n^\delta}} \times \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ with $\delta < \epsilon$ and $d \in N$ such that, if r is the truth table of an n^δ variable boolean function $f : \{0,1\}^{n^\delta} \to \{0,1\}$ with $S^A(f_{n^\delta}) > n^d$, then the function $G(s) = F(r,s)$ is an A-PRG.

Proof For any $\delta > 0$, applying hardness amplification to an f satisfying the above hardness condition for sufficiently large d results in a function g such that $H_g^A(n^\delta) > kn^2$ for some constant k. For any ϵ, by a relativization of Theorem 2.5, the NW generator $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ based on g is an A-PRG if δ is chosen sufficiently small. By Theorem 2.6, the design for this

generator can be constructed in time polynomial in n. Computing G requires evaluating f on poly(n) inputs. This can be done in time 2^{n^δ} for each input with access to the truth table of f, by looking up the values.

Theorem 2.39 (Theorem 14 [10]) Let A be any oracle. For every $\epsilon > 0$, there is a polynomial time computable function $F : \{0,1\}^{n^c} \times \{0,1\}^{d \log(n)} \to \{0,1\}^n$ with $c, d \in N$ such that, if r is the truth table of a $c \log(n)$-variable Boolean function $f : \{0,1\}^{c \log n} \to \{0,1\}$ with $S^A\left(f_{c \log n}\right) > n^{\epsilon c}$, then $G_r\left(s\right) = F\left(r, s\right)$ is an A-PRG.

The proof of Theorem 2.39 is similar to that of Theorem 2.38 and is omitted.

Chapter 3

Derandomization and Probabilistic Polynomial Hierarchies

This chapter contains the original contributions of the thesis, which include a number of results concerning the hierarchies MA^{Σ_i} and AM^{Σ_i} introduced in Section 1.2. Section 3.1 will derive an equivalence between circuit lower bounds and derandomization for MA^{Σ_i}. This is a generalization of a result of Impagliazzo et al. concerning MA [10]. Using this result and a diagonalization argument of Kannan, MA^{Σ_i} will be shown to separate from NEXP^{Σ_i} at almost all levels in Section 3.2. Section 3.3 uses the same diagonalization to prove a tradeoff between derandomizations of BPP and of AM^{Σ_i}. Lu and Impagliazzo et. al. prove various tradeoffs between deterministic simulations of nondeterministic time and derandomizations of AM [20][10]. Section 3.4 will derive similar tradeoffs for the hierarchy AM^{Σ_i}.

3.1 MA^{Σ_i} and the Exponential Hierarchy

The previous chapter derived a set of circuit lower bound conditions for uniformly computable functions sufficient to derandomize BPP to various degrees (Theorem 2.34). It is not known if such lower bounds are also a necessary condition for a derandomization of BPP. However, in [12], it is shown that a specific problem in coRP, Polynomial Identity Testing, cannot be simulated in $\cap_{\epsilon>0}\mathrm{NTIME}\left(2^{n^\epsilon}\right)$ unless either NEXP does not have polynomial size circuits or the permanent of a matrix cannot be computed by polynomial size arithmetic circuits. Thus, even a mild derandomization of BPP requires polynomial size circuit lower bounds.

In the case of the classes AM and MA of Arthur-Merlin games, it has been proven that derandomization is equivalent to a polynomial size circuit lower bound for NEXP [10]. Before stating this result formally, we first introduce a notation.

Definition 3.1 For any complexity class C, C/poly denotes the class of languages decidable in C with polynomial advice. That is, $L \in$C/poly if there exist $L' \in$C and an advice string $h(n) \in \{0,1\}^{p(n)}$ for some polynomial p such that $x \in L$ iff $x \cdot h(|x|) \in L'$, where $x \cdot h(|x|)$ denotes x concatenated with $h(|x|)$. More generally, C/$f(n)$ denotes the class of languages decidable in C with $f(n)$ advice.

Theorem 3.2 P/poly is the class of languages decidable by polynomial size circuits. More generally, P^A/poly is the class of languages decidable by polynomial size A-oracle circuits.

Proof Suppose a language L is decidable by polynomial size circuits. Then L is decidable by a polynomial time Turing Machine taking the circuit description as advice, which simulates the circuit on its input and returns the answer. Conversely, suppose $L \in$P/poly. Then L can be decided by hardwiring the advice string into the circuit that simulates the polynomial time Turing Machine for L. The general case follows from a relativization of this proof.

The following theorem is proved by Impagliazzo et al. [10].

Theorem 3.3 NEXP$\subseteq$P/poly $\leftrightarrow$ MA = NEXP.

The backward direction of Theorem 3.3 is proved using the hardness-randomness tradeoffs of Chapter 2. The forward direction is an extension of the following result of Babai et al. [2].

Theorem 3.4 EXP$\subseteq$ P/poly $\rightarrow$ MA = EXP.

In this section, we extend Theorem 3.3 to show the following result about the hierarchy MA^{Σ_i}:

Theorem 3.5 $\mathrm{NEXP}^{\Sigma_i} \subseteq \mathrm{P}^{\Sigma_i}$/poly $\leftrightarrow \mathrm{MA}^{\Sigma_i} = \mathrm{NEXP}^{\Sigma_i}$, $i \geq 0$.

The proof of this theorem follows closely that of Theorem 3.3 [10]. As a consequence of Theorem 3.5, we derive an unconditional derandomization of almost all levels of MA^{Σ_i} in the next section.

Babai et al. prove that EXP can be simulated by multi-prover interactive proof systems. In a multi-prover interactive protocol, a set of all-powerful machines $\{P_1, P_2, ...P_k\}$ (the provers) exchange messages with a probabilistic polynomial time machine V (the verifier), attempting to convince the verifier to accept an input string x. The provers are not allowed to communicate with each other, and cannot read the verifier's random bits.

Definition 3.6 A language L has a multi-prover interactive proof system (MIPS) if there is a multi-prover interactive protocol such that

1. If $x \in L$ then there exist (honest) provers $\{P_1, P_2, ...P_k\}$ such that $\Pr(V\text{accepts}) > 1 - 2^{-n}$.

2. If $x \notin L$ then for all sets of provers $\{P'_1, P'_2, ...P'_k\}$, $\Pr(V\text{accepts}) < 2^{-n}$.

EXP can be simulated by a particular type of MIPS with restrictions on the power of the provers.

Definition 3.7 Let C be a complexity class. A language L has a multi-prover interactive proof system of complexity C if L has a multi-prover interactive proof system such that each honest prover P_i is restricted to answering membership questions for a language $L_i \in$ C.

Theorem 3.8 [2] For any $L \in$ EXP, L has a multi-prover interactive proof system of complexity EXP.

This property of EXP implies Theorem 3.4. We require a generalization of this theorem.

Theorem 3.9 If EXP $\subseteq \mathrm{P}^{\Sigma_i}$/poly, then EXP $= \mathrm{MA}^{\Sigma_i}$, $i \geq 0$.

Proof Take $L \in$ EXP. By Theorem 3.8, if EXP $\subseteq \mathrm{P}^{\Sigma_i}$/poly, L has a MIPS such that the honest provers are Σ_i-oracle circuits of polynomial size. The following is a 2 round Arthur-Merlin protocol for L that uses an oracle for Σ_i. On input x, Merlin gives Arthur a set of Σ_i-oracle circuits and Arthur uses them to verify in probabilistic polynomial time with the help of the oracle that $x \in L$. If $x \notin L$, then, from Definition 3.6, x will be rejected with high probability regardless of what circuits were supplied. If $x \in L$, then,

for some set of circuits, x will be accepted with high probability. Therefore, $L \in \mathrm{MA}^{\Sigma_i}$.

The following unconditional and implied separations and inclusions are essentially taken from Impagliazzo et al. [10]. They are adapted to the current context in some cases by relativizing some of the complexity classes to Σ_i. In these cases, we give the necessary adjustments to the proofs of the theorems from which they were adapted. We indicate the corresponding theorem in each case.

Definition 3.10 For any oracle A, let $\mathrm{SIZE}^A(f(n))$ denote the set of languages decidable by A-oracle circuits of size at most $f(n)$.

Theorem 3.11 (Theorem 2 [10]) For any fixed $c \in N$, EXP $\not\subseteq$ io-$\mathrm{SIZE}^{\Sigma_i}(n^c)$, $i \geq 0$.

Proof The following proof follows that of Theorem 3 in Impagliazzo et al. [10]. The set S of all Σ_i oracle circuits of size at most n^c has size at most $2^{kn^{2c}}$ for some constant k. This is because there are at most 5^{n^c} subsets of gates, and for each gate, there are n^{2c} ways of connecting its two output lines, giving an upper bound of $5^{n^c} \cdot n^{2cn^c} = 5^{n^c} \cdot 2^{2c\log(n)n^c} \in 2^{O(n^{2c})}$ on the number of circuits. Define L to be the language accepted by the following deterministic Turing machine M. On input x of length n, M rejects if x is not one of the first $kn^{2c}+1$ strings of length n in lexicographic order. Otherwise, let $\{x_1, x_2, ...x_i\}$ be the lexicographically-ordered sequence of the first i strings of length n with $x = x_i$. Define the set $\{S_0, S_1, S_2, ...S_i\}$ of subsets of S as follows: $S_0 = S$, and for $1 \leq j \leq i$, S_j is the set of circuits in S_{j-1} that reject x_j if more than half of the circuits in S_{j-1} accept x_j, otherwise it is the set of all circuits in S_{j-1} that accept x_j. M finds the sets $S_1, ...S_i$ by evaluating the appropriate circuits on the inputs $x_1, ...x_i$. If S_i is empty, M rejects x. Otherwise, it accepts x if S_i is the set of all circuits in S_{i-1} that reject x and rejects x otherwise. Clearly, $L \notin$ io-$\mathrm{SIZE}^{\Sigma_i}(n^c)$. Since the number of circuits evaluated by M is exponential and each circuit can be evaluated in exponential time, $L \in$ EXP.

Theorem 3.12 (Theorem 3 [10]) For any $c \in N$, EXP $\not\subseteq$ io-$\mathrm{DTIME}\left(2^{n^c}\right)/n^c$.

Proof The proof is the same as that of Theorem 3.11, except that the set S is now the set of Turing machines that take n^c advice and have description size at most n. Since, for sufficiently large n, any Turing machine has

a description of length less than n, the algorithm diagonalizes against all machines with the specified time bound and advice for all but finitely many n. For the details of the proof, see Impagliazzo et al. [10].

Theorem 3.13 [10] If $\mathrm{NEXP}^{\Sigma_{i-1}} \subseteq \mathrm{P}^{\Sigma_{i-1}}/\mathrm{poly}$ then $\mathrm{NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n \subseteq \mathrm{SIZE}^{\Sigma_{i-1}}\left(n^{d_0}\right)$ for some constant d_0, $i \geq 1$.

Proof This is Lemma 5 of Impagliazzo et al. relativized to Σ_i. Their proof involves reducing a language decidable by a nondeterministic machine with advice to a language in NEXP decided by a universal Turing machine which, by assumption, is decidable by a polynomial size circuit. This proof can be extended to the present case by replacing nondeterministic machines with alternating Turing machines.

Theorem 3.14 [10] If $\mathrm{NEXP}^{\Sigma_{i-1}} = \mathrm{EXP}$, then

$$\mathrm{NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n \subseteq \mathrm{DTIME}\left(2^{n^{d_0}}\right)/n$$

for some constant d_0, $i \geq 1$.

Proof This is Lemma 6 of Impagliazzo et. al. with the nondeterministic complexity classes relativized to Σ_i. The proof is similar to that of Theorem 3.13.

Theorem 3.15 (Corollary 8 [10]) If $\mathrm{NEXP}^{\Sigma_{i-1}} \subseteq \mathrm{P}^{\Sigma_{i-1}}/\mathrm{poly}$, then $\mathrm{EXP} \not\subseteq \mathrm{io} - \mathrm{NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n$, $i \geq 1$.

Proof By Theorem 3.13, the hypothesis implies

$$\mathrm{NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n \subseteq \mathrm{SIZE}^{\Sigma_{i-1}}\left(n^{d_0}\right)$$

for some constant d_0. If $\mathrm{EXP} \subseteq \text{io-NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n$, then $\mathrm{EXP} \subseteq \mathrm{SIZE}^{\Sigma_{i-1}}\left(n^{d_0}\right)$, contradicting Theorem 3.11.

Theorem 3.16 (Corollary 9 [10]) If $\mathrm{NEXP}^{\Sigma_{i-1}} = \mathrm{EXP}$, then $\mathrm{NEXP}^{\Sigma_{i-1}} \not\subseteq \text{io-NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n$, $i \geq 1$.

Proof The hypothesis implies, by Theorem 3.14, that $\mathrm{NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n \subseteq \mathrm{DTIME}\left(2^{n^{d_0}}\right)/n$ for some constant d_0. If $\mathrm{NEXP}^{\Sigma_{i-1}} \subseteq \text{io-NTIME}\left(2^n\right)^{\Sigma_{i-1}}/n$

$\subseteq$ io-DTIME $\left(2^{n^{d_0}}\right)/n$, then EXP $\subseteq$ io-DTIME $\left(2^{n^{d_0}}\right)/n$, contradicting Theorem 3.12.

We will also need the following result.

Theorem 3.21 [4] If $\mathrm{EXP}^{\Sigma_i} \subseteq \mathrm{EXP/poly}$ then $\mathrm{EXP}^{\Sigma_i} = \mathrm{EXP}$, $i \geq 0$.

The following results concern the derandomization of the hierarchy MA^{Σ_i}. These theorems are proved in Impagliazzo et al. for the corresponding non-relativized complexity classes [10].

Theorem 3.22 (Theorem 12 [10]) For any $i \geq 0$, suppose there is an algorithm A in $\mathrm{NTIME}\left(2^{O(n)}\right)^{\Sigma_i}$ that, given 1^n as input and $a(n)$ advice, outputs on every accepting computational path the truth table of a function $f : \{0,1\}^n \to \{0,1\}$ such that, for all $d \in N$, $S^{\Sigma_i}(f_n) > n^d$ i.o. Then, for every $\epsilon > 0$,

$$\mathrm{MA}^{\Sigma_i} \in \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}/a\left(n^\epsilon\right) \tag{3.1}$$

Proof Take $L \in \mathrm{MA}^{\Sigma_i}$. By a relativization of Theorem 1.4, there exists a relation $R \in \mathrm{P}^{\Sigma_i}$ such that

$$x \in L \leftrightarrow \exists y \in \{0,1\}^m \left[\Pr_{z\in\{0,1\}^p}\left[R\left(x,y,z\right)=1\right] \geq \frac{2}{3}\right]$$
$$x \notin L \leftrightarrow \forall y \in \{0,1\}^m \left[\Pr_{z\in\{0,1\}^p}\left[R\left(x,y,z\right)=1\right] \leq \frac{1}{3}\right]$$

for some m and p which are polynomial in $|x|$. Allowing unused random bits if necessary, p can be chosen to be the polynomial time bound for deciding R. By Theorem 2.38, for every $\epsilon > 0$, for infinitely many n, a Σ_i-PRG $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^p$ can be computed in time 2^{n^ϵ} given a truth table of length 2^{n^δ} for some $\delta < \epsilon$ which is output by A on an accepting path. The following nondeterministic Σ_i-oracle machine M decides L for infinitely many input lengths. On input x of length n, M runs A on 1^{n^δ} using the appropriate advice. If A rejects, M rejects. Otherwise, M guesses the witness y and computes $R\left(x,y,z\right)$ for each z in the range of G, where G is the Σ_i-PRG constructed from the output of A, accepting iff $R\left(x,y,z\right)=1$ for the majority of z. Clearly, M runs in $\mathrm{NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}/a(n^\epsilon)$.

Theorem 3.23 [10] For $i \geq 0$, if EXP $\not\subseteq \mathrm{P}^{\Sigma_i}/\mathrm{poly}$, then $\forall\ \epsilon > 0$,

$$\mathrm{MA}^{\Sigma_i} \subseteq \bigcap_{\epsilon>0} \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i} \tag{3.2}$$

Proof Let $f \in$ EXP be a function that cannot be simulated by polynomial size Σ_i oracle circuits. Then f satisfies condition 1) of Theorem 2.37 with $\mathrm{A} = \Sigma_i$ for infinitely many input lengths. The "infinitely often" form of Theorem 2.37 implies the existence of a generator $G : \{0,1\}^{n^\epsilon} \rightarrow \{0,1\}^n$ that can be used by a nondeterministic Σ_i-oracle machine to carry out the required simulation for infinitely many n. In the notation of the proof of Theorem 3.22, this machine guesses y and accepts if the probability taken over all outputs z of G that $R(x,y,z) = 1$ is greater than $\frac{1}{2}$. This can be checked in time 2^{n^ϵ}.

Definition 3.24 [10] For a relation $R(x,y)$ with $x \in \{0,1\}^n$ and $y \in \{0,1\}^{2^n}$, $n \in N$, let $f_R(x)$ denote the boolean function such that

$$f_R(x) = 1 \leftrightarrow \exists y \in \{0,1\}^{2^n} [R(x,y) = 1]$$

Let $f_{R,A,s}$ denote the boolean function such that

$$f_{R,A,s}(x) = 1 \leftrightarrow \exists y \in T_{A,s}(|x|) [R(x,y) = 1]$$

where $T_{A,s}(|x|)$ is the set of all truth tables of boolean functions f on $|x|$ bits such that $S^A(f) < s$.

Theorem 3.25 (Lemma 17 [10]) For $i \geq 0$, suppose there exists a DTIME $\left(2^{O(n)}\right)^{\Sigma_i}$ relation $R(x,y)$ with $x \in \{0,1\}^n$ and $y \in \{0,1\}^{2^n}$, $n \in N$, such that $f_R(x) \neq f_{R,A,s}(x)$ for some $x \in \{0,1\}^n$ for infinitely many n, where A is any oracle. Then there is a NTIME $\left(2^{O(n)}\right)^{\Sigma_i}$ algorithm that, given advice of length n, for infinitely many n, nondeterministically generates on every accepting computational path the truth table of a boolean function g on n variables with $S^A(g) > s$.

Proof If $f_R(x) \neq f_{R,A,s}(x)$ then $f_R(x) = 1$ and $f_{R,A,s}(x) = 0$. Given x as advice, the algorithm guesses y and accepts and outputs y iff $R(x,y) = 1$.

Theorem 3.26 [10] For $i \geq 0$, if $\mathrm{NEXP}^{\Sigma_i} \neq \mathrm{EXP}^{\Sigma_i}$, then there exists a DTIME $\left(2^{O(n)}\right)^{\Sigma_i}$ relation $R(x,y)$ with $x \in \{0,1\}^n$ and $y \in \{0,1\}^{2^n}$, such

that for every fixed $d \in \mathrm{N}$, for infinitely many n, there is a $x \in \{0,1\}^n$ such that $f_R(x) \neq f_{R,\Sigma_i,n^d}(x)$.

Proof Suppose that for every such relation there is a $d \in \mathrm{N}$ such that, for all but finitely many n and for $x \in \{0,1\}^n$, $f_R(x) = f_{R,\Sigma_i,n^d}(x)$. Take $L \in \mathrm{NEXP}^{\Sigma_i}$ and let 2^{n^k} be the time bound of the nondeterministic Σ_i oracle machine that decides L. Consider the language L' defined as follows: for each word of length n in L, L' contains this word padded with 0's to length n^k. Then there is a relation R as defined in the statement of the Theorem such that $f_R(x) = 1$ iff $x \in L'$ and $f_R(x) = f_{R,\Sigma_i,n^d}(x)$ for all but finitely many n for some d. Thus, L' can be decided in EXP^{Σ_i} by computing, on input x, the truth tables, y, of all Σ_i oracle circuits on n inputs of size at most n^d and checking if $R(x,y) = 1$ for at least one such y. Since the time bound for deciding L' is exponential in the length of the original unpadded strings, and since L can be reduced to L' in polynomial time, $L \in \mathrm{EXP}^{\Sigma_i}$. Thus $\mathrm{NEXP}^{\Sigma_i} = \mathrm{EXP}^{\Sigma_i}$.

Theorem 3.27 [10] If $\mathrm{NEXP}^{\Sigma_i} \neq \mathrm{EXP}^{\Sigma_i}$, then, for every $\epsilon > 0$, $\mathrm{MA}^{\Sigma_i} \subseteq \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}/n^\epsilon$.

Proof This follows from Theorems 3.22, 3.25 and 3.26.

Proof (of Theorem 3.5) ($\rightarrow$) Suppose

$$\mathrm{NEXP}^{\Sigma_i} \subseteq \mathrm{P}^{\Sigma_i}/\mathrm{poly} \tag{3.3}$$

but

$$\mathrm{NEXP}^{\Sigma_i} \neq \mathrm{EXP} \tag{3.4}$$

Inclusion (3.4) implies, from Theorem 3.9, that $\mathrm{EXP} = \mathrm{MA}^{\Sigma_i}$. It also implies $\mathrm{EXP}^{\Sigma_i} \subseteq \mathrm{NEXP}^{\Sigma_i} \subseteq \mathrm{P}^{\Sigma_i}/\mathrm{poly} \subseteq \mathrm{EXP}/\mathrm{poly}$, so by Theorem 3.21, $\mathrm{EXP}^{\Sigma_i} = \mathrm{EXP}$. Therefore, from (3.5), $\mathrm{NEXP}^{\Sigma_i} \neq \mathrm{EXP}^{\Sigma_i}$. By Theorem 3.27, $\forall \epsilon > 0$

$$\begin{aligned} \mathrm{MA}^{\Sigma_i} &\subseteq \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}/n^\epsilon \\ \mathrm{EXP} &\subseteq \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}/n^\epsilon \\ &\subseteq \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}/n^\epsilon \end{aligned} \tag{3.5}$$

Inclusions (3.4) and (3.6) contradict Theorem 3.15. Therefore, (3.4) implies $\mathrm{NEXP}^{\Sigma_i} = \mathrm{EXP} = \mathrm{MA}^{\Sigma_i}$.

($\leftarrow$) Suppose

$$\mathrm{NEXP}^{\Sigma_i} = \mathrm{MA}^{\Sigma_i} \tag{3.6}$$

but

$$\mathrm{NEXP}^{\Sigma_i} \not\subseteq \mathrm{P}^{\Sigma_i}/\mathrm{poly} \tag{3.7}$$

Assumption (3.7) implies that $\mathrm{NEXP}^{\Sigma_i} = \mathrm{EXP}$, so $\mathrm{EXP} \not\subseteq \mathrm{P}^{\Sigma_i}/\mathrm{poly}$ by (3.8). By Theorem 3.23, this implies

$$\begin{aligned} \mathrm{MA}^{\Sigma_i} &\subseteq \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i} \\ \mathrm{NEXP}^{\Sigma_i} &\subseteq \text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i} \\ &\subseteq \text{io-NTIME}\left(2^n\right)^{\Sigma_i}/n \end{aligned}$$

This contradicts Theorem 3.16, so the theorem is proved.

3.2 Unconditional Derandomization of MA^{Σ_i}

Theorem 3.3 suggests that finding a nontrivial derandomization of MA is a difficult problem that will require a new, nonrelativizable proof method. This is because there is an oracle A such that $\mathrm{NEXP}^A \subseteq \mathrm{P}^A/\mathrm{poly}$ [8]. However, relativizable methods do suffice to prove the lower bound of Theorem 3.5 for nearly all i, leading to the following separation result.

Theorem 3.28 For all but at most one i, $\mathrm{MA}^{\Sigma_i} \neq \mathrm{NEXP}^{\Sigma_i}$, $i \geq 0$.

In [14], a diagonzalization argument is used to prove that Σ_i, $i \geq 2$, does not have n^k size circuits for any fixed k. To prove the lower bound required for Theorem 3.28, we use a scaled-up version of this argument to show that the exponential hierarchy $\Sigma_i^{\mathrm{EXP}} = \mathrm{NEXP}^{\Sigma_i}$ defined in section 1.2 has an exponential size circuit lower bound for $i \geq 4$. This is proved by exhibiting an exponential time ATM that, for any given input length, simulates a circuit with no equivalent circuit below a certain exponential size. Before describing this simulation, we show that such a circuit exists.

Theorem 3.29 For any $\epsilon > 0$, there is a circuit with n inputs of size 2^{2n} which accepts a subset of $\{0,1\}^n$ not accepted by any circuit of size $2^{(1-\epsilon)n}$.

Proof The set $\{0,1\}^n$ of strings of length n has 2^{2^n} subsets. Let $s_n(f)$ denote the number of circuits on n inputs of size at most f. We show that, for any $\epsilon > 0$, $s_n\left(2^{(1-\epsilon)n}\right)$ is $o\left(2^{2^n}\right)$ and so at least one subset of strings of length n is not accepted by any such circuit for sufficiently large n. For any $\epsilon > 0$, the number of possible sets of gates for a circuit of size at most $2^{(1-\epsilon)n}$ is bounded above by $4^{2^{(1-\epsilon)n}}$. Since each gate has two output lines, the number of ways each gate's output lines can be connected is at most $2^{2(1-\epsilon)n}$. Therefore, there are at most $2^{2(1-\epsilon)n2^{(1-\epsilon)n}}$ ways of connecting the gates. So,

$$\begin{aligned} s_n\left(2^{(1-\epsilon)n}\right) &\leq 4^{2^{(1-\epsilon)n}} 2^{2(1-\epsilon)n2^{(1-\epsilon)n}} \\ &= 2^{2\cdot 2^{(1-\epsilon)n} + 2(1-\epsilon)n2^{(1-\epsilon)n}} \\ &= 2^{[2+2(1-\epsilon)n]2^{(1-\epsilon)n}} \\ &\in O\left(2^{2^{(1-\epsilon')n}}\right) \quad \text{for some } 0 < \epsilon' < 1 \end{aligned}$$

This shows that there is a subset, S, of $\{0,1\}^n$ that is not accepted by a $2^{(1-\epsilon)n}$-size circuit for sufficiently large n. On the other hand, each string in S is accepted by a circuit of size $2n$, and there are at most 2^n strings in S. Since a circuit accepting S can be obtained by ORing together the circuits that accept each string, S is accepted by a circuit of size $2n2^n + 2^n \leq 2^{2n}$.

Theorem 3.30 For any $\epsilon > 0$, there is a function $f \in \Sigma_4^{\text{EXP}} \bigcap \Pi_4^{\text{EXP}}$ with $S(f) \geq 2^{(1-\epsilon)n}$.

Proof Kannan proves that there is a function $f \in \Sigma_4 \bigcap \Pi_4$ that does not have circuits of size $O(n^k)$ for any given k by constructing a polynomial length QBF that simulates a circuit of size n^{2k+5} that does not have an equivalent size n^{k+1} circuit [14]. The proof of Theorem 3.30 is identical except that the QBF is of exponential length and simulates a circuit of size 2^{2n} that has no equivalent size $2^{(1-\epsilon)n}$ circuit.

Corollary 3.31 $\Sigma_2^{\text{EXP}} \bigcap \Pi_2^{\text{EXP}} \not\subseteq \text{P/poly}$.

Proof If NP $\subseteq$ P/poly, then $\Sigma_i = \Sigma_2$, $\forall\ i \geq 2$ [15], and by a padding argument, this implies $\Sigma_i^{\text{EXP}} = \Sigma_2^{\text{EXP}}$, $\forall\ i \geq 2$. The result then follows

from Theorem 3.30. If NP $\not\subseteq$ P/poly, the result follows from the fact that NP $\subseteq \Sigma_2^{\mathrm{EXP}} \bigcap \Pi_2^{\mathrm{EXP}}$.

Theorem 3.28 now follows from Theorem 3.5 and the following Theorem of [4].

Theorem 3.32 [4] There is at most one $i \geq 0$ such that $\mathrm{NEXP}^{\Sigma_i} \subseteq \mathrm{P}^{\Sigma_i}/\mathrm{poly}$.

Proof Suppose there were two numbers $j \geq 0$ and $k \geq 0$ with $j < k$ such that $\mathrm{NEXP}^{\Sigma_j} \subseteq \mathrm{P}^{\Sigma_j}/\mathrm{poly}$ and $\mathrm{NEXP}^{\Sigma_k} \subseteq \mathrm{P}^{\Sigma_k}/\mathrm{poly}$. By Theorem 3.5, $\mathrm{NEXP}^{\Sigma_j} = \mathrm{NEXP}^{\Sigma_k} = \mathrm{EXP}$, so $\mathrm{NEXP}^{\Sigma_k} \subseteq \mathrm{P}^{\Sigma_j}/\mathrm{poly}$. The proofs of Theorem 3.30 and Corollary 3.31 relativize, and relativizing Corollary 3.31 to Σ_j implies that $\left(\Sigma_2^{\mathrm{EXP}}\right)^{\Sigma_j} = \mathrm{NEXP}^{\mathrm{NP}^{\Sigma_j}} = \mathrm{NEXP}^{\Sigma_{j+1}} = \Sigma_{j+2}^{\mathrm{EXP}} \not\subseteq \mathrm{P}^{\Sigma_j}/\mathrm{poly}$. But this implies $\mathrm{NEXP}^{\Sigma_k} \not\subseteq \mathrm{P}^{\Sigma_j}/\mathrm{poly}$ since $\Sigma_{j+2}^{\mathrm{EXP}} \subseteq \Sigma_{k+1}^{\mathrm{EXP}} \equiv \mathrm{NEXP}^{\Sigma_k}$, a contradiction.

3.3 A Derandomization Tradeoff

While the lower bound NEXP $\not\subseteq$ P/poly implies a separation of NEXP and MA by Theorem 3.3, no explicit nontrivial upper bound has been shown to follow from this lower bound. A stronger derandomization of AM (and of MA) follows from a circuit lower bound for NEXP$\bigcap$co-NEXP. Klivans and van Melkebeek prove a spectrum of hardness-randomness tradeoffs analogous to Theorem 2.34 for AM [17]. These tradeoffs are similar to those for BPP except with circuit lower bounds for deterministic functions replaced by NP-oracle circuit lower bounds for functions in NEXP$\bigcap$co-NEXP. The proof uses an NW generator G based on a function $f \in \mathrm{NEXP} \bigcap \mathrm{co\text{-}NEXP}$ satisfying such a lower bound. By the relativized hardness-randomness tradeoffs of section 2.2.3, G is an NP-PRG, and the i^{th} bit of G can be computed nondeterministically by guessing a witness that certifies acceptance or nonacceptance of f on its input. It follows from Theorem 1.4 that G allows a nontrivial nondeterministic simulation of AM.

In this section, we generalize the tradeoffs of Klivans and van Melkebeek, deriving a set of hardness-randomness tradeoffs for $\mathrm{AM}^{\Sigma_{i-1}} = \mathrm{BP} \cdot \Sigma_i$. We also derive an interesting tradeoff between derandomizations of BPP and of $\mathrm{BP} \cdot \Sigma_i$ that follows from the hardness randomness tradeoffs. The following Theorem establishes a Σ_i-oracle circuit lower bound that implies a complete

derandomization of $\mathrm{BP}\cdot\Sigma_i$, i.e. $\mathrm{BP}\cdot\Sigma_i=\Sigma_i$. It is analogous to Theorem 3.13 of Klivans and van Melkebeek [17].

Theorem 3.33 Let Σ_i^{E} and Π_i^{E} denote the $2^{O(n)}$-time analogues of Σ_i and Π_i, respectively. For $i\geq 1$, if there exists a function $f\in\Sigma_i^{\mathrm{E}}\cap\Pi_i^{\mathrm{E}}$ with $S^{\Sigma_i}(f)\in 2^{\Omega(n)}$, then $\mathrm{BP}\cdot\Sigma_i=\Sigma_i$.

Proof Take $L\in\mathrm{BP}\cdot\Sigma_i$. By Theorem 1.9, there is a relation $M\in\mathrm{P}$ such that, for $|x|=p$ and $n=\mathrm{poly}\,(p)$,

$$\begin{aligned} x\in L &\leftrightarrow \forall y\in\{0,1\}^n\left[\exists z_1\forall z_2\cdots Q_i z_i\left(M\left(x,z_1,...z_i,y\right)=1\right)\right] \\ x\notin L &\leftrightarrow \Pr_{y\in\{0,1\}^n}\left[\exists z_1\forall z_2\cdots Q_i z_i\left(M\left(x,z_1,...z_i,y\right)=1\right)\right]\leq\frac{1}{3}\end{aligned} \tag{3.8}$$

Allowing unused random bits if necessary, n can be chosen to be the minimum size of a circuit computing M. Let f be a function as in Theorem 3.33, and let ϵ be a constant such that $S^{\Sigma_i}(f)>2^{\epsilon n}$. Then $S^{\Sigma_i}\left(f_{c\log(n)}\right)\geq n^{\epsilon c}$ a.e., so there exist $c,d\in N$ and a function F as in Theorem 2.39 such that $G(s)=F(r,s)$ is a Σ_i-PRG $G:\{0,1\}^{d\log n}\rightarrow\{0,1\}^n$ if r is the truth table of f on inputs of length $c\log n$. Therefore (3.9) holds with $\{0,1\}^n$ replaced by the range of G. Since $f\in\Sigma_i^{\mathrm{E}}\cap\Pi_i^{\mathrm{E}}$, there are $2^{O(|x|)}$-time decidable relations R_0 and R_1 such that

$$\begin{aligned} f\left(x\right)=0 &\leftrightarrow \exists u_1\forall u_2\cdots Q_i u_i\left[R_0\left(x,u_1,u_2,...u_i\right)=1\right] \\ f\left(x\right)=1 &\leftrightarrow \exists u_1\forall u_2\cdots Q_i u_i\left[R_1\left(x,u_1,u_2,...u_i\right)=1\right]\end{aligned}$$

for some $2^{O(|x|)}$ length u_1, u_2,...u_i. Let $r\left(j\right)$ denote the j^{th} bit of string r, and let t_j and s_j denote the j^{th} string in $\{0,1\}^{c\log(n)}$ and $\{0,1\}^{d\log(n)}$, respectively, in lexicographic order. Then,

$$\begin{aligned} &x\in L\leftrightarrow\exists r\in\{0,1\}^{n^c}\left[\wedge_{t_j}\exists u_1^j\forall u_2^j\cdots Q_i u_i^j\left(R_{r(j)}\left(t_j,u_1^j,u_2^j,...u_i^j\right)=1\right)\right. \\ &\left.\wedge_{s_j}\exists z_1^j\cdots Q_i z_i^j\left(M\left(x,z_1^j,...z_i^j,G\left(s_j\right)\right)=1\right)\right] \\ &\leftrightarrow\exists r\exists z_1^1\cdots\exists z_1^{n^d}\exists u_1^1\cdots\exists u_1^{n^c}\forall z_2^1\cdots\forall z_2^{n^d}\forall u_2^1\cdots\forall u_2^{n^c}\cdots \\ &Q_i z_i^1\cdots Q_i z_i^{n^d}Q_i u_i^1\cdots Q_i u_i^{n^c}\left[\left(\wedge_{t_j}R_{r(j)}\left(t_j,u_1^j,u_2^j,...u_i^j\right)=1\right)\right. \\ &\left.\wedge_{s_j}\left(M\left(x,z_1^j,...z_i^j,G\left(s_j\right)\right)=1\right)\right]\end{aligned} \tag{3.9}$$

The expression in square brackets in (3.10) is decidable in time polynomial in n. Therefore, $L\in\Sigma_i$.

The lower bound of Theorem 3.33 cannot be proved using diagonalization, since it does not relativize [16]. However, if it is assumed that $\Sigma_i \subseteq \mathrm{P/poly}$, then $\mathrm{P}^{\Sigma_i}/\mathrm{poly}$ is the same as P/poly, in which case the diagonalization method of the previous section suffices to show $\mathrm{BP} \cdot \Sigma_i = \Sigma_i$. If this assumption is false, then $\mathrm{EXP} \not\subseteq \mathrm{P/poly}$, which implies nontrivial derandomizations of BPP and MA.

Theorem 3.34 If $\Sigma_i \subseteq \mathrm{P/poly}$, then there is a function $f \in \Sigma_4^{\mathrm{E}} \bigcap \Pi_4^{\mathrm{E}}$ such that $S^{\Sigma_i}(f) > 2^{(1-\epsilon)n}$ for some $\epsilon > 0$.

Proof If $\Sigma_i \subseteq \mathrm{P/poly}$ then, in a Σ_i-oracle circuit of size $2^{(1-\epsilon)n}$, the Σ_i-oracle gates can be replaced by ordinary circuits of size at most $2^{k(1-\epsilon)n}$ for some k because each gate has at most $2^{(1-\epsilon)n}$ inputs. Take ϵ sufficiently large that $k(1-\epsilon) + (1-\epsilon) = \epsilon'$ for some $0 < \epsilon' < 1$. A Σ_i-oracle circuit of size $2^{(1-\epsilon)n}$ can be computed by an ordinary circuit of size $2^{\epsilon' n}$. The result now follows from Theorem 3.30.

Theorem 3.35 [15] If $\mathrm{NP} \subseteq \mathrm{P/poly}$, then $\Sigma_i = \Sigma_2$, $i \geq 2$.

Theorem 3.36 Either $\mathrm{BP} \cdot \Sigma_i = \Sigma_i \ \forall i \geq 2$, or $\mathrm{BPP} \subseteq \bigcap_{\epsilon \geq 0} \text{io-DTIME}\left(2^{n^\epsilon}\right)$ and $\mathrm{MA} \subseteq \bigcap_{\epsilon \geq 0} \text{io-NTIME}\left(2^{n^\epsilon}\right)$.

Proof If $\Sigma_i \subseteq \mathrm{P/poly}$, then $\mathrm{NP} \subseteq \mathrm{P/poly}$, which implies $\Sigma_i = \Sigma_2 \ \forall i \geq 2$ by Theorem 3.35. A padding argument shows that this implies $\Sigma_i^{\mathrm{E}} = \Sigma_2^{\mathrm{E}}$ $\forall i \geq 2$, so by Theorem 3.34, there is an $f \in \Sigma_2^{\mathrm{E}} \cap \Pi_2^{\mathrm{E}}$ with a Σ_i-oracle circuit lower bound of $2^{\Omega(n)}$. By Theorem 3.33, this means $\mathrm{BP} \cdot \Sigma_i = \Sigma_i$ for $i \geq 2$. If $\Sigma_i \not\subseteq \mathrm{P/poly}$, then $\mathrm{EXP} \not\subseteq \mathrm{P/poly}$, implying $\mathrm{BPP} \subseteq \bigcap_{\epsilon \geq 0} \text{io-DTIME}\left(2^{n^\epsilon}\right)$ by the io form of Theorem 2.34 and $\mathrm{MA} \subseteq \bigcap_{\epsilon > 0} \text{io-NTIME}\left(2^{n^\epsilon}\right)$ by Theorem 3.23.

Other tradeoffs can be derived by, for example, increasing the upper bound on Σ_i. This involves a corresponding decrease in the lower bound on the function f of Theorem 3.34, since there is a greater increase in size when converting from a Σ_i-oracle circuit to an ordinary circuit. The resulting simulation of $\mathrm{BP} \cdot \Sigma_i$ is less efficient, because there are more elements in the range of the generator that must be tested for acceptance. Some of the tradeoffs that can be proven are shown in Table 3.1. Note that, in cases in which there is no upper bound of P/poly for Σ_i, the corresponding tradeoff holds only for $i \geq 4$, since Theorem 3.35 does not apply.

Upper Bound for Σ_i-complete function f	Implied Lower Bound for $f \in \Sigma_4^E \bigcap \Pi_4^E$	Tradeoff
$S(f) \leq n^k$ for some k	$S^{\Sigma_i}(f) \in 2^{\Omega(n)}$	Either BP $\cdot \Sigma_i = \Sigma_i$, $i \geq 2$ or BPP$\subseteq$ $\bigcap_{\epsilon>0} \left[\text{io-DTIME}\left(2^{n^\epsilon}\right)\right]$
$S(f) \leq 2^{n^\epsilon}$ for any $\epsilon > 0$	$S^{\Sigma_i}(f) \geq n^k$ for any k	Either BP $\cdot \Sigma_i \subseteq$ $\bigcap_{\epsilon>0} \text{NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}$, $i \geq 4$ or BPP $\subseteq$ io-DTIME $\left(2^{(\log n)^c}\right)$ for some c
$S(f_n) \leq n^k$ io for some k	$S^{\Sigma_i}(f_n) \geq 2^{(1-\epsilon)n}$ for some $\epsilon > 0$, io	Either BP$\cdot\Sigma_i \subseteq \text{io} - \Sigma_i$, $i \geq 4$ or BPP $\subseteq \bigcap_{\epsilon>0} \text{DTIME}\left(2^{n^\epsilon}\right)$
$S(f_n) \leq 2^{n^\epsilon}$ io for any $\epsilon > 0$	$S^{\Sigma_i}(f_n) \geq n^k$, io for any k	Either BP$\cdot\Sigma_i \subseteq$ $\bigcap_{\epsilon>0} \left[\text{io-NTIME}\left(2^{n^\epsilon}\right)^{\Sigma_i}\right]$, $i \geq 4$ or BPP $\subseteq$ DTIME $\left(2^{(\log n)^c}\right)$, for some c

Table 3.1: Derandomization tradeoffs

3.4 Randomness and Nondeterminism

The proof of Theorem 3.27 is an example of the "easy witness method", a general technique invented by Kabanets for proving tradeoffs between derandomizations of randomized complexity classes and deterministic simulations of nondeterministic time [13]. The idea of the method is that, if every word in a nondeterministically decidable language has a witness that is "easy" in the sense that it is the truth table of a small circuit, then the language can be decided efficiently by a deterministic algorithm by searching over all small circuits for one whose truth table is the witness for the input. If there is a nondeterministically decidable language for which this is not the case, then, given a word that has only "hard" witnesses, the truth tables of functions of high circuit complexity can be generated nondeterministically and used to carry out a nontrivial derandomization of a randomized complexity class via results such as Theorems 2.38 and 2.39.

The easy witness method has been used to prove a number of tradeoffs involving AM and MA. This has led to several interesting results in derandomization, including the derandomization - circuit lower bound equivalence of Theorem 3.3, a gap theorem for zero-error probabilistic exponential time, and an upper bound for graph nonisomorphism [10][20]. In this section, we apply the easy witness method to the hierarchy BP·Σ_i, proving several tradeoffs that generalize those proved by Lu and Impagliazzo et. al. for AM [10][20].

The derandomization algorithm that will be used for BP·Σ_i is set out in the following theorem, which is analogous to Theorem 3.22.

Theorem 3.37 Suppose there is a nondeterministic exponential time algorithm A that, on input 1^n and given $a(n)$ advice, outputs on every accepting computational path the truth table of a function $f : \{0,1\}^p \rightarrow \{0,1\}$, $p > n$ with $S^{\Sigma_i}(f_n) \in 2^{\Omega(n)}$. Then, for $i \geq 1$, $\mathrm{BP} \cdot \Sigma_i = \Sigma_i/a(c \log n)$ for some constant c.

Proof Take $L \in \mathrm{BP} \cdot \Sigma_i$. By Theorem 1.9, there is a relation $M \in \mathrm{P}$ such that, for $m = \mathrm{poly}(|x|)$,

$$\begin{aligned} x \quad &\in L \leftrightarrow \forall y \in \{0,1\}^m \left[\exists z_1 \forall z_2 \cdots Q_i z_i M\left(x, y, z_1, z_2, ...z_i\right)\right] \qquad (3.10) \\ x \quad &\notin L \leftrightarrow Pr_{y \in \{0,1\}^m}\left[\exists z_1 \forall z_2 \cdots Q_i z_i M\left(x, y, z_1, z_2, ...z_i\right)\right] \leq \frac{1}{3} \end{aligned}$$

where $Q_i = \exists$ ($\forall$) for odd (even) i. For any fixed x, the predicate

$$\exists z_1 \forall z_2 \cdots Q_i z_i M\left(x, y, z_1, z_2, ...z_i\right)$$

is computable by a function $f(y)$ with $S^{\Sigma_i}(f)$ a polynomial in n. Allowing unused random bits if necessary, m can be assumed to be equal to $S^{\Sigma_i}\left(f_{|x|}\right)$. By Theorem 2.39, for sufficiently large n, for some $c, d \in N$, a Σ_i-PRG $G : \{0,1\}^{d \log n} \to \{0,1\}^m$ can be computed in time n^c given a truth table of length n^c output by A on an accepting computation. Let $S = \{g_1, g_2, ...g_{|G|}\}$ be the range of G. Then,

$$\begin{aligned}
x \in L &\leftrightarrow \forall y \in G\left[\exists z_1 \forall z_2 \cdots Q_i z_i M\left(x, y, z_1, z_2, ...z_i\right)\right] \\
&\leftrightarrow \exists z_1^1 \forall z_2^1 \cdots Q_i z_i^1 M\left(x, g_1, z_1^1, z_2^1, ...z_i^1\right) \wedge \\
&\quad \exists z_1^2 \forall z_2^2 \cdots Q_i z_i^2 M\left(x, g_2, z_1^2, z_2^2, ...z_i^2\right) \wedge \\
&\quad \cdots \exists z_1^{|G|} \forall z_2^{|G|} \cdots Q_i z_i^{|G|} M\left(x, g_{|G|}, z_1^{|G|}, z_2^{|G|}, ...z_i^{|G|}\right) \\
&\leftrightarrow \exists z_1^1 \exists z_1^2 \cdots \exists z_1^{|G|} \forall z_2^1 \forall z_2^2 \cdots \forall z_2^{|G|} \cdots Q_i z_i^1 Q_i z_i^2 \cdots Q_i z_i^{|G|} \\
&\quad \left[M\left(x, g_1, z_1^1, z_2^1, ...z_i^1\right) \wedge M\left(x, g_2, z_1^2, z_2^2, ...z_i^2\right) \wedge \right. \\
&\quad \left. \cdots M\left(x, g_{|G|}, z_1^{|G|}, z_2^{|G|}, ...z_i^{|G|}\right)\right] \qquad (3.11)
\end{aligned}$$

The following ATM M decides L in $\Sigma_i/a(c \log n)$. On input x of length n, M runs A on $1^{c \log n}$ using the appropriate advice. If A rejects, M rejects. Otherwise, M constructs G using the output of A and evaluates the predicate in (3.12).

We now show that the algorithm A in Theorem 3.37 exists unless there is a subexponential time deterministic simulation of NP for which it is in some sense difficult to find inputs on which the simulation fails.

Definition 3.38 A nondeterministic refuter is a nondeterministic algorithm that, on input 1^n, outputs a string of length n and either accepts or rejects on each nondeterministic branch.

Definition 3.39 A nondeterministic refuter distinguishes two languages L and L' for length n if, on input 1^n, whenever the refuter accepts, it outputs a string in $L \Delta L'$, where $L \Delta L'$ is the symmetric difference of L and L'.

Definition 3.40 For any complexity class C, [pseudo$_{\text{FNP}}$] − C ([io-pseudo$_{\text{FNP}}$] − C) is the set of languages L such that there is a language

$L' \in C$ such that any polynomial time nondeterministic refuter fails to distinguish L and L' for length n a.e. (i.o.).

The following theorem generalizes Theorem 3.1 of Lu [20].

Theorem 3.41 Either NP $\subseteq \bigcap_{\epsilon>0} [\text{io-pseudo}_{\text{FNP}}]\,\text{DTIME}\left(2^{n^\epsilon}\right)$ or $\text{BP} \cdot \Sigma_i = \Sigma_i$ for $i \geq 1$.

Proof Any language in NP is decidable by a function $f_M(x)$, where

$$f_M(x) = 1 \leftrightarrow \exists y \in \{0,1\}^m \, [M(x,y) = 1] \tag{3.12}$$

for some polynomial time decidable predicate $M(x,y)$ where $m = \text{poly}(|x|)$. Define $T_{A,s}(n)$ as in Definition 3.24, and define $f_{M,A,s}(x)$ as

$$f_{M,A,s}(x) = 1 \leftrightarrow \exists y \in T_{A,s}(\log m) \, [M(x,y) = 1] \tag{3.13}$$

For any function $f \in$ NP, define a deterministic simulation D that, instead of evaluating $f_M(x)$, evaluates f_{M,Σ_i,m^δ} for some constant δ by searching over all circuits of size at most m^δ. The set $T_{\Sigma_i,m^\delta}(\log m)$ contains at most $2^{m^{2\delta}}$ truth tables. Since an Σ_i-oracle gate in a circuit of size m^δ can be evaluated in time $2^{O(m^\delta)}$, each truth table can be generated in time $2^{O(m^\delta)}$. Therefore, the simulation runs in DTIME($2^{m^{c\delta}}$) for some constant c. For any ϵ, δ can be chosen so that the simulation runs in DTIME(2^{n^ϵ}).

If NP $\not\subseteq \bigcap_{\epsilon>0}$[io-pseudo$_{FNP}$]DTIME($2^{n^\epsilon}$), there must be a nondeterministic refuter that, on input 1^n, outputs on every accepting computation a string x of length n for which $f_M(x) \neq f_{M,\Sigma_i,m^\delta}(x)$ for some $f_M \in$ NP and $\delta > 0$, a.e. Given x, the truth table, y, of a function f' with $S^{\Sigma_i}\left(f'_{\log m}\right) \geq m^\delta = 2^{\delta \log m}$, can be generated nondeterministically by guessing y and checking $M(x,y) = 1$. Since this takes nondeterministic polynomial time, by Theorem 3.37, $\text{BP} \cdot \Sigma_i = \Sigma_i$.

Other types of tradeoffs are possible which differ from Theorem 3.41 in the way in which the algorithm that derandomizes $\text{BP} \cdot \Sigma_i$ is provided with the string on which the deterministic simulation of nondeterministic time fails. In Theorem 3.41, this was done using the nondeterministic refuter. Another possibility is to provide this information as advice, as was done in the proof of Theorem 3.27. This gives the following variant.

Theorem 3.42 Either NP $\subseteq \bigcap_{\epsilon>0}$ io-DTIME $\left(2^{n^\epsilon}\right)$ or BP $\cdot \Sigma_i \subseteq \Sigma_i/\text{poly}$ for $i \geq 1$.

Proof If NP $\not\subseteq \bigcap_{\epsilon>0}$ io-DTIME $\left(2^{n^\epsilon}\right)$, then there exists $x \in \{0,1\}^n$ a.e. such that $f_M(x) \neq f_{M,\Sigma_i,m^\delta}(x)$ for some $\delta > 0$, where f_M and f_{M,Σ_i,m^δ} are defined as in the proof of Theorem 3.41. Define a nondeterministic algorithm A that on input $1^{\log n}$ and given x as advice, guesses a witness and accepts and outputs this witness iff it is a witness for x. This witness is the truth table of a function f on $\log m$ bits with $S^{\Sigma_i}(f_{\log m}) \geq m^\delta$, so by Theorem 3.37, BP $\cdot \Sigma_i = \Sigma_i/2^{c \log n} = \Sigma_i/n^c$ for some constant c.

Using the concatenation of the witnesses for all possible input strings as the source of hardness eliminates the necessity of knowing the specific string that has a hard witness. This method is used to derive the following tradeoff involving only standard uniform complexity classes, which generalizes Theorem 19 of Impagliazzo et. al. [10].

Theorem 3.43 Either NE $\bigcap$ coNE $\subseteq \bigcap_{\epsilon>0}$ io-DTIME$(2^{2^{\epsilon n}})$, or BP $\cdot \Sigma_i = \Sigma_i$ for $i \geq 1$.

Proof For any $L \in$ NE $\cap$ coNE, there are DTIME $\left(2^{O(n)}\right)$ decidable relations M_+ and M_- and $m = 2^{O(n)}$ such that for $x \in \{0,1\}^n$,

$$\begin{aligned} x &\in L \leftrightarrow \exists y \in \{0,1\}^m [M_+(x,y) = 1] \\ x &\notin L \leftrightarrow \exists y \in \{0,1\}^m [M_-(x,y) = 1] \end{aligned}$$

Define $f_{M_+}(x)$ and $f_{M_+,A,s}(x)$ as in the proof of Theorem 3.41. If NE $\bigcap$ coNE $\not\subseteq$ $\bigcap_{\epsilon>0}$ io-DTIME $\left(2^{2^{\epsilon n}}\right)$, then there is an $x \in \{0,1\}^n$ a.e. such that $f_{M_+}(x) \neq f_{M_+,\Sigma_i,2^{\epsilon n}}(x)$. Take $\{0,1\}^n = \{x_1, x_2, ...x_{2^n}\}$ and let $\{y_1, ...y_{2^n}\} \subseteq \{0,1\}^{2^n}$ be a set of strings such that $M_+(x_i, y_i) = 1$ or $M_-(x_i, y_i) = 1$ for all $1 \leq i \leq 2^n$. Let Y be the concatenation of the strings $\{y_1, ...y_{2^n}\}$. Then Y can be computed in nondeterministic time $2^{O(n)}$ and is the truth table of a 2n variable function f' with $S^{\Sigma_i}(f'_{2n}) > 2^{\epsilon n}$. The conclusion now follows from Theorem 3.37.

Chapter 4

Conclusion

This thesis investigated the derandomization of two probabilistic polynomial hierarchies, MA^{Σ_i} and AM^{Σ_i}. The main results are the equivalence of circuit lower bounds and derandomization of MA^{Σ_i} (Theorem 3.5), the separation of MA^{Σ_i} and EH at almost all levels (Theorem 3.28), and a set of tradeoffs between derandomizations of AM^{Σ_i} and of BPP (Table 3.1).

There are many possibilities for further research in this area. The results presented here suggest that derandomizations of BPP, MA, and AM could be proven indirectly by investigating the relationship between MA^{Σ_i}, AM^{Σ_i}, and EH. It may also be useful to study further the relation between circuit lower bounds and derandomization. In particular, one could attempt to find a version of Theorem 3.5 involving AM^{Σ_i}, or to extend this theorem by proving a range of derandomization - lower bound equivalences analogous to the range of hardness - randomness tradeoffs in Theorem 2.34.

Appendix A

Proof of Theorem 1.4

We first express a k-round Arthur-Merlin game, denoted AM[k] or MA[k], as a quantified boolean formula with k alternating blocks of quantifiers. We then prove that this alternating hierarchy collapses. In [26], the following notation is used to characterize randomized complexity classes.

Definition A.1 [26] Let $\exists^+ z\,(M)$, where M is any quantified boolean expression and z is a string of boolean variables, denote $\Pr_z\,(M) \geq \frac{2}{3}$. Then $(Q_1 Q_2 \cdots Q_k / Q'_1 Q'_2 \cdots Q'_k)$, where, for $1 \leq i \leq k$, $Q_i, Q'_i \in \{\exists, \forall, \exists^+\}$, is the class of languages L such that there exists a polynomial-time decidable predicate M with

$$\begin{aligned} x \quad &\in L \leftrightarrow Q_1 z_1 Q_2 z_2 \cdots Q_k z_k\,(M) \\ x \quad &\notin L \leftrightarrow Q'_1 z'_1 Q'_2 z'_2 \cdots Q'_k z'_k\,(\neg M) \end{aligned} \tag{A.1}$$

where, for example, $Q_i z_i$ denotes a string of quantified boolean variables with quantifier Q_i. For this set of languages to be nonempty, the quantifiers Q_i and Q'_i must satisfy

$$Q'_1 z'_1 Q'_2 z'_2 \cdots Q'_k z'_k\,(\neg M) \rightarrow \neg Q_1 z_1 Q_2 z_2 \cdots Q_k z_k\,(M) \tag{A.2}$$

Theorem A.2 [26] Let AM[k] (MA[k]) denote the set of languages decidable by Arthur-Merlin games with k rounds in which Arthur (Merlin) makes the first move. Then

$$\text{AM[k]} \;=\; \left(\exists^+\exists\exists^+ \cdots / \exists^+\forall\exists^+ \cdots\right) \tag{A.3}$$

$$\text{MA[k]} \;=\; \left(\exists\exists^+\exists \cdots / \forall\exists^+\forall \cdots\right) \tag{A.4}$$

where each string of quantifiers in (A.3) and (A.4) contains k quantifiers.

The proof of Theorem A.2 requires a lemma.

Lemma A.3 [26] For $L \in$ AM (MA), there is an Arthur-Merlin game in which Arthur's (Merlin's) turn comes first such that, for some polynomial $q(n)$,

1) For $x \in L$, Merlin can convince Arthur to accept x with probability at least $1 - 2^{-q(n)}$.

2) For $x \notin L$, Merlin cannot convince Arthur to accept with probability greater than $2^{-q(n)}$.

Proof The proof is similar to that of Theorem 1.1. Since $L \in$AM, there is a k-round Arthur-Merlin game G satisfying the two properties listed in Definition 1.2. Consider the k-round Arthur-Merlin game G' in which Arthur and Merlin play t copies of the game G in parallel with t given by (1.4). This is done by performing, in each round of G', the computations of the corresponding round of G separately for each of t different game histories for G. Merlin wins the game G' iff Merlin wins for more than half of the t parallel games. G' satisfies 1) and 2) of Lemma A.3 by the same argument as in the proof of Theorem 1.1.

Proof (of Theorem A.2) The case $k = 2$ is straightforward. For example, for AM[2] the theorem states that, if for the majority of two-round games Merlin wins (loses), then for the majority of moves of Arthur, Merlin has (does not have) a winning move.

In general, for any language $L \in$AM[k] (or MA[k]), there is a k-round Arthur-Merlin game G such that, for most sequences of moves of Arthur and Merlin, Merlin wins on input $x \in L$ and loses on input $x \notin L$. To prove Theorem A.2, we must show that there is an Arthur-Merlin game G' with most moves of Arthur *in each round in which it is Arthur's turn* leading to acceptance for $x \in L$ and rejection for $x \notin L$. Let $s(n)$ be the number of possible sequences of moves of Arthur in G for $n = |x|$ and let $u(n)$ be the minimum number of possible moves of Arthur in any given Arthur round of G. Let G' be a game for L satisfying the conditions of Lemma A.3 for some $q(n)$ to be determined. From the proof of Lemma A.3, Arthur has $ts(n)$ possible sequences of moves in G' and at least $tu(n)$ possible moves in any given round of G', for t given by (1.4). To ensure that the fraction of possible moves in any round of G' leading to a correct decision concerning membership of x in L is at least $\frac{2}{3}$, $q(n)$ must be chosen so that the number of possible moves in any Arthur round of G' must be at least 3 times the number of

sequences of moves of Arthur in G' that lead to an incorrect decision. Thus, we choose $q(n)$ so that

$$\begin{aligned} \frac{tu(n)}{ts(n)} &> 3 \cdot 2^{-q(n)} \\ \frac{u(n)}{3s(n)} &> 2^{-q(n)} \\ \log \frac{u(n)}{3s(n)} &> -q(n) \\ q(n) &> \log \frac{3s(n)}{u(n)} \end{aligned} \tag{A.5}$$

Since $s(n)$ and $u(n)$ are polynomial in n, $q(n)$ can be chosen to be $O(\log n)$. For this choice, G' runs in time polynomial in n, so the theorem is proved.

The next theorem shows that the $\forall$ and $\exists^+$ quantifiers can be swapped in a QBF.

Theorem A.4 [26] For any polynomial time decidable predicate P,

$$\forall y \exists^+ z P(x, y, z) \rightarrow \exists^+ C \forall y \exists z \in C P(x, y, z) \tag{A.6}$$

where y and z are polynomial length strings of boolean variables and C is a set of strings of length $|z| = f(n)$ of size $g(n)$, $n = |x|$, for some polynomials f and g.

Proof Take $g(n) = f(n) + 3$. The probability taken over all sets C that $\neg \forall y \exists z \in C [P(x, y, z)]$ satisfies

$$\begin{aligned} &\Pr_C [\exists y \forall z \in C \neg (P(x, y, z))] \\ &\leq \sum_{y \in \{0,1\}^{f(n)}} \Pr_C [\forall z \in C (\neg P(x, y, z))] \\ &\leq \sum_{y \in \{0,1\}^{f(n)}} \left(\frac{1}{3}\right)^{g(n)} \\ &= 2^{f(n)} \left(\frac{1}{3}\right)^{f(n)+3} \\ &\leq \frac{1}{3} \end{aligned}$$

The theorem follows.

Lemma A.5 [25] For P a polynomial time decidable predicate,

$$\forall u \exists^{+} v P(x,u,v) \rightarrow \forall C \exists^{+} v \wedge_{u \in C} P(x,u,v) \tag{A.7}$$

where C is a set of strings of length $|u| = f(n)$ of size $g(n)$, $n = |x|$.

Proof The following proof is from Zachos and Heller (Lemma 4 [25]). From Theorem 1.1, it can be assumed that $\forall u \left[\Pr_v \left(P(x,u,v)\right) \geq 1 - 2^{-p(n)}\right]$ for some polynomial p. For any C,

$$\begin{aligned} \Pr_v \left[\vee_{u \in C} \neg P(x,u,v)\right] &\leq \sum_{u \in C} \Pr_v \left[\neg P(x,u,v)\right] \\ &\leq g(n) \frac{1}{2^{p(n)}} \\ &\leq \frac{1}{3} \end{aligned}$$

where the last upper bound holds for sufficiently large n. This proves the lemma.

Theorem A.6 [25] $(\exists^{+}/\exists^{+}) = (\exists^{+}\forall/\forall\exists^{+}) = (\forall\exists^{+}/\exists^{+}\forall)$

Proof We follow the proof of Zachos and Heller (Theorem 5 [25]). Take $L \in (\exists^{+}/\exists^{+})$, and let P be a polynomial time decidable predicate such that $x \in L \leftrightarrow \exists^{+} u P(x,u)$ and $x \notin L \leftrightarrow \exists^{+} u \neg P(x,u)$, where $|u| = p(|x|)$. Then,

$$\begin{aligned} x \in L &\rightarrow \exists^{+} u P(x,u) \\ &\rightarrow \forall s \exists^{+} u P\left(x, (u+s) \bmod 2^{p(|x|)}\right), \text{ for } s \in \{0,1\}^{p(|x|)} \\ &\rightarrow \exists^{+} C \forall s \exists u \in C P\left(x, (u+s) \bmod 2^{p(|x|)}\right), \text{ by (A.6)} \\ &\rightarrow \exists^{+} C \forall s \vee_{u \in C} P\left(x, (u+s) \bmod 2^{p(|x|)}\right) \end{aligned} \tag{A.8}$$

Also,

$$\begin{aligned} x \notin L &\rightarrow \exists^{+} u \neg P(x,u) \\ &\rightarrow \forall s \exists^{+} u \neg P\left(x, (u+s) \bmod 2^{p(|x|)}\right), \text{ for } s \in \{0,1\}^{p(|x|)} \\ &\rightarrow \forall C \exists^{+} u \wedge_{s \in C} \neg P\left(x, (u+s) \bmod 2^{p(|x|)}\right), \text{ by (A.7)} \\ &\rightarrow \forall C \exists^{+} u \neg \vee_{s \in C} P\left(x, (u+s) \bmod 2^{p(|x|)}\right) \end{aligned} \tag{A.9}$$

For any fixed set C of strings, the disjuncts in (A.8) and (A.9) are negations of each other. Also, since C is of polynomial size, these disjuncts can be decided in polynomial time. Therefore, $L \in (\exists^+\forall/\forall\exists^+)$.

Conversely, take $L \in (\exists^+\forall/\forall\exists^+)$, and let P be a polynomial time decidable predicate such that $x \in L \rightarrow \exists^+ u \forall v P(x,u,v)$ and $x \notin L \rightarrow \forall u \exists^+ v \neg P(x,u,v)$. Then,

$$\begin{aligned} x &\in L \rightarrow \exists^+ (u,v)\, P(x,u,v) \\ x &\notin L \rightarrow \exists^+ (u,v)\, \neg P(x,u,v) \end{aligned}$$

Therefore, $L \in (\exists^+/\exists^+)$. The equality $(\exists^+\forall/\forall\exists^+) = (\forall\exists^+/\exists^+\forall)$ follows from the fact that $(\exists^+/\exists^+)$ is closed under complement.

Theorem A.7 [26] $(\exists\forall/\forall\exists^+) \subseteq (\forall\exists/\exists^+\forall)$.

Proof The following proof is from Zachos (Theorem 1 [26]). Take $L \in (\exists\forall/\forall\exists^+)$. Then, for some polynomial time decidable predicate P,

$$\begin{aligned} x \notin L &\rightarrow \forall y \exists^+ z \neg P(x,y,z) && \text{(A.10)} \\ &\rightarrow \exists^+ C \forall y \exists z \in C \neg P(x,y,z), \text{ by Theorem A.4} && \text{(A.11)} \\ &\rightarrow \exists C \forall y \exists z \in C \neg P(x,y,z) && \text{(A.12)} \\ &\rightarrow \forall y \exists z \neg P(x,y,z) && \text{(A.13)} \\ &\rightarrow \neg \exists y \forall z P(x,y,z) && \text{(A.14)} \\ &\rightarrow x \notin L && \text{(A.15)} \end{aligned}$$

This shows that the implied expressions in (A.10) to (A.15) are equivalent. From (A.11)

$$\begin{aligned} x \notin L &\leftrightarrow \exists^+ C \forall y \exists z \in C \neg P(x,y,z) \\ &\leftrightarrow \exists^+ C \forall y \neg [\forall z \in C (P(x,y,z))] \end{aligned}$$

and from (A.12),

$$\begin{aligned} x \notin L &\leftrightarrow \exists C \forall y \exists z \in C \neg P(x,y,z) \\ x \notin L &\leftrightarrow \neg \forall C \exists y \forall z \in C [P(x,y,z)] \\ x \in L &\leftrightarrow \forall C \exists y \forall z \in C [P(x,y,z)] \end{aligned}$$

Since $|C|$ is a polynomial, $\forall z \in C [P(x,y,z)]$ is decidable in polynomial time. Therefore, $L \in (\forall\exists/\exists^+\forall)$.

Remark Theorems A.4, A.6, and A.7 generalize to the case where the involved quantifiers are embedded in a larger string of quantifiers. For example, Theorem A.6 generalizes to $(Q_1\exists^+Q_2/Q_3\exists^+Q_4) = (Q_1\exists^+\forall Q_2/Q_3\forall\exists^+Q_4)$, where Q_1, Q_2, Q_3, and Q_4 are strings of quantifiers. These generalizations can be proved using the fact that quantifiers distribute over conjunction and disjunction.

Theorem A.8 [26] i) $(\exists\exists^+/\forall\exists^+) = (\exists\forall/\forall\exists^+)$
ii) $(\exists^+\exists/\exists^+\forall) = (\forall\exists/\exists^+\forall)$

Proof i)

$$\begin{aligned} & \left(\exists\exists^+/\forall\exists^+\right) \\ = & \left(\exists\exists^+\forall/\forall\forall\exists^+\right), \text{ by Theorem A.6} \\ \subseteq & \left(\exists\exists\forall/\forall\forall\exists^+\right) \\ = & \left(\exists\forall/\forall\exists^+\right) \end{aligned}$$

ii)

$$\begin{aligned} & \left(\exists^+\exists/\exists^+\forall\right) \\ = & \left(\forall\exists^+\exists/\exists^+\forall\forall\right), \text{ by Theorem A.6} \\ \subseteq & \left(\forall\exists\exists/\exists^+\forall\forall\right) \\ = & \left(\forall\exists/\exists^+\forall\right) \end{aligned}$$

The reverse inclusions are obvious.

Proof (of Theorem 1.4) The fact that MA = NP·BP follows from Theorem A.2, since NP·BP = $(\exists\exists^+/\forall\exists^+)$. To show that AM = BP·NP = $(\exists^+\exists/\exists^+\forall)$, it is sufficient to show that AM$[k+1]$ = AM$[k]$ for all $k \geq 2$. For $k = 3$,

$$\begin{aligned} \text{AM}[3] &= \left(\exists^+\exists\exists^+/\exists^+\forall\exists^+\right) \\ &= \left(\exists^+\exists\exists^+\forall/\exists^+\forall\forall\exists^+\right), \text{ by Theorem A.6} \\ &\subseteq \left(\exists^+\exists\forall/\exists^+\forall\exists^+\right) \\ &\subseteq \left(\exists^+\forall\exists/\exists^+\exists^+\forall\right), \text{ by Theorem A.7} \end{aligned}$$

$$
\begin{aligned}
&= \left(\exists^+\forall\forall\exists/\forall\exists^+\exists^+\forall\right), \text{ by Theorem A.6} \\
&= \left(\exists^+\forall\exists/\forall\exists^+\forall\right) \\
&\subseteq \left(\exists\forall\exists/\forall\exists^+\forall\right) \\
&\subseteq \left(\forall\exists\exists/\exists^+\forall\forall\right), \text{ by Theorem A.7} \\
&= \left(\forall\exists/\exists^+\forall\right) \\
&= \left(\exists^+\exists/\exists^+\forall\right), \text{ by Theorem A.8 ii)} \\
&= \mathrm{AM}[2]
\end{aligned}
$$

It follows from the above and the remark following the proof of Theorem A.7 that $\mathrm{AM}[k+1] = \mathrm{AM}[k]$. Therefore, $\mathrm{AM}[k] = \mathrm{BP}\cdot\mathrm{NP}\ \forall\ k$.

Bibliography

[1] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. "BPP has Subexponential Time Simulations unless EXPTIME has Publishable Proofs," *Computational Complexity* **3** (1993), 307-318.

[2] L. Babai, L. Fortnow, and C. Lund. "Non-deterministic Exponential Time has Two-prover Interactive Protocols," *Computational Complexity* **1**(1) (1991), 3-40.

[3] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity.* Toronto: Prentice Hall, 1994.

[4] H. Buhrman, L. Fortnow, and A. Pavan. "Some Results on Derandomization," *Lecture Notes in Computer Science* **2607** (2003), 212-222.

[5] H. Buhrman and S. Homer. "Superpolynomial Circuits, almost Sparse Oracles and the Exponential Hierarchy," *Lecture Notes in Computer Science* **652** (1992), 116-127.

[6] D. Du and K. Ko. *Theory of Computational Complexity.* Toronto: John Wiley and Sons, 2000.

[7] O. Goldreich and L. A. Levin. "A Hard-core Predicate for all One-way Functions," in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing* (1989), 25-32.

[8] H. Heller. "On Relativized Exponential and Probabilistic Complexity Classes," *Information and Control* **71** (1986), 231-243.

[9] R. Impagliazzo. "Hard-core Distributions for Somewhat Hard Problems," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science* (1995), 538-545.

[10] R. Impagliazzo, V. Kabanets, and A. Wigderson. "In Search of an Easy Witness: Exponential Time vs. Probabilistic Polynomial Time," *Journal of Computer and System Sciences* **65** (2002), 672-694.

[11] R. Impagliazzo and A. Wigderson. "P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), 220-229.

[12] V. Kabanets and R. Impagliazzo. "Derandomizing Polynomial Identity Tests means Proving Circuit Lower Bounds," *Computational Complexity* **13** (2004), 1-46.

[13] V. Kabanets. "Easiness Assumptions and Hardness Tests: Trading Time for Zero Error," *Journal of Computer and System Sciences* **63**(2) (2001), 236-252.

[14] R. Kannan. "Circuit-size Lower Bounds and Non-reducibility to Sparse Sets," *Information and Control* **55** (1982), 40-56.

[15] R. M. Karp and R. J. Lipton. "Some Connections between Nonuniform and Uniform Complexity Classes," in *Proceedings of Twelfth Annual ACM Symposium on Theory of Computing* (1980), 302-309.

[16] K. Ko. "Separating and Collapsing Results on the Relativized Probabilistic Polynomial Hierarchy," *Journal of the ACM* **37**(2) (1990), 415-438.

[17] A. Klivans and D. van Melkebeek. "Graph Nonisomorphism has Subexponential Size Proofs unless the Polynomial-time Hierarchy collapses," *SIAM Journal on Computing* **31**(5) (2002), 1501-1526.

[18] R. J. Lipton. "New Directions in Testing," in J. Feigenbaum and M. Merritt, editors, *Distributed Computing and Cryptography*, DIMACS Volume 2 (1989), 191-202.

[19] L. A. Levin. "One-Way Functions and Pseudorandom Generators," *Combinatorica* **7**(4) (1987), 357-363.

[20] C. Lu. "Derandomizing Arthur-Merlin Games under Uniform Assumptions," *Computational Complexity* **10**(3) (2001), 247-259.

[21] M. Mitzenmacher and E. Upfal. *Probability and Computing.* New York: Cambridge University Press, 2005.

[22] N. Nisan and A. Wigderson. "Hardness vs. Randomness," *Journal of Computer and System Sciences* **49** (1994), 149-167.

[23] M. Sipser. *Introdution to the Theory of Computation.* Toronto: PWS Publishing Company, 1997.

[24] A. C. Yao. "Theory and Application of Trapdoor Functions," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science* (1982), 80-91.

[25] S. Zachos and H. Heller. "A Decisive Characterization of BPP," *Information and Control* **69** (1986), 125-135.

[26] S. Zachos and M. Furer. "Probabilistic Quantifiers vs. Distrustful Adversaries," *Lecture Notes in Computer Science* **287** (1987), 443-455.