P=BPP unless E has sub-exponential circuits: Derandomizing the XOR Lemma

(Preliminary Version)

Russell Impagliazzo* — russell@cs.ucsd.edu
Avi Wigderson† — avi@cs.huji.ac.il

November 9, 1996

Abstract

Yao showed that the XOR of independent random instances of a somewhat hard Boolean function becomes almost completely unpredictable. In this paper we show that, in non-uniform settings, total independence is not necessary for this result to hold. We give a pseudo-random generator which produces n instances of the function for which the analog of the XOR lemma holds. This is the first derandomization of a "direct product" result. Our generator is a combination of two known ones - the random walks on expander graphs of [1, 9, 19] and the nearly disjoint subsets generator of [23]. The quality of the generator is proved via a new proof of the XOR lemma, which might also be useful for other direct product results.

Combining our generator with the approach of [25, 6] and the generator of [16] gives substantially improved results for hardness vs. randomness trade-offs. In particular, we show that if any problem in $E = DTIME(2^{O(n)})$ has circuit complexity $2^{\Omega(n)}$, then $P = BPP$.

*Computer Science and Engineering, UC San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0114. Research supported by NSF YI Award CCR-92-570979, Sloan Research Fellowship BR-3311, grant #93025 of the joint US-Czechoslovak Science and Technology Program, and USA-Israel BSF Grant 92-00043

†Institute of Computer Science, The Hebrew University, Jerusalem, Israel 91904. Part of this work was done while on sabbatical leave at the Institute for Advanced Study, Princeton and Princeton University. Research supported by the Sloan Foundation, American-Israeli BSF grant 92-00106, and the Wolfson Research Awards, administered by the Israel Academy of Sciences.

1 Introduction

This paper addresses the relationship between three central questions in complexity theory. First, to what extent can a problem be easier to solve for probabilistic algorithms than for deterministic ones? Secondly, what properties should a pseudo-random generator have so that its outputs are "random enough" for the purpose of simulating a randomized algorithm? Thirdly, if solving one instance of a problem is computationally difficult, is solving several instances of the problem proportionately harder?

Yao's seminal paper ([30]) was the first to show that these questions are related. Building on work by Blum and Micali ([7]) he showed how to build, from any cryptographically secure one-way permutation, a pseudo-random generator whose outputs are indistinguishable from truly random strings to any reasonably fast computational method. He then showed that such a psuedo-random generator could be used to deterministically simulate any probabilistic algorithm with sub-exponential overhead.

Blum, Micali, and Yao thus showed the central connection between "hardness" (the computational difficulty of a problem) and "randomness" (the utility of the problem as the basis for a pseudo-random generator to de-randomize computation).

A completely different approach for trading hardness and randomness was introduced by Nisan and Wigderson [25]. It allows hardness assumptions for much "harder" functions, namely functions in EXP, deterministic exponential time. The best such trade-off, appearing in [6], shows that if any function in EXP requires circuit size $2^{n^{\Omega(1)}}$, then $BPP \subseteq \tilde{P}$ (with $\tilde{P} = DTIME(2^{(\log n)^{O(1)}})$. In words, under the natural hardness assumption above, randomness is just slightly useful in speeding computation, in that it can always be simulated deterministically with quasi-polynomial slow-down.

The main consequence of our work is a significantly improved trade-off: if some function in $E = DTIME(2^{O(n)})$ has worst-case circuit complexity $2^{\Omega(n)}$, then $BPP = P$. In other words, randomness *never* speeds computation by more than a polynomial amount unless non-uniformity *always* helps computation more than polynomially for problems with exponential time complexity. This adds to a number of results showing which show that $P = BPP$ follows either from a "hardness" result or an "easiness" result. On the one hand, there is a sequence of "hardness" assumptions implying $P = BPP$ ([25], [4],[5]; this last, in work done independently from this paper, draws the somewhat weaker conclusion that $P = BPP$ follows from the existence of a function in E with circuit complexity $\Omega(2^n/n)$.) On the other hand, $P = BPP$ follows from the easiness premise $P = NP$[29],or even the weaker statement $E = EH$ [6] (where EH is the alternating hierarchy above E). We feel that from these results, the evidence favors using as a working conjecture that $P = BPP$. However, it has not even been proved unconditionally that $BPP \neq NE$! This is a sad state of affairs, and one we hope will be rectified soon. One way our result could help do so is if there were some way of "certifying" random Boolean functions as having circuit complexity $2^{\Omega(n)}$. (See [28] for some discussion of this possibility.)

The source of this improvement is in the key feature common to both general trade-offs of [30] and of [25] - the amplification of the hardness of a function. In both one needs to convert a mildly hard function into one that is nearly unpredictable by circuits of a given size. The tool for such amplification was provided in the same paper of Yao, and became to be known as Yao's XOR-Lemma. The XOR Lemma can be paraphrased as follows: Fix a non-uniform model of computation (with certain closure properties) and a boolean function $f : \{0,1\}^n \to \{0,1\}$. Assume that any algorithm in the model of a certain complexity has a significant probability of failure when predicting f on a randomly chosen instance x. Then any algorithm (of a slightly smaller complexity) that tries to guess the XOR $f(x_1) \oplus f(x_2) \oplus \cdots \oplus f(x_k)$ of k random instances $x_1, \cdots, x_k$ won't do significantly better than a random coin toss.

The main source of "slack" in the best previous trade-offs is the k fold increase of input size from the mildly hard function to the unpredictable one, resulting from the independence of the k instances. In this paper, we show that true independence of the instances is not necessary for a version of the XOR Lemma to hold. Our main technical contribution is a way to pick k (pseudo-random) instances $x_1, \cdots, x_k$ of a somewhat hard problem, using many fewer that kn random bits, but for which the XOR of the bits $f(x_i)$ is still almost as hard to predict as if the instances were independent. Indeed, for the parameters required by the aforementioned application, we use only $O(n)$ random bits to decrease the prediction to exponentially small amount. Combining this with previous techniques yileds the trade-off above.

Realizing that our task is basically a derandomization task should be comforting. Derandomization has become a major subfield of theoretical computer science, with a wealth of results and techniques. The general recipie requires a careful study of the use of stochastic independence in the probabilistic solution, often revealing "structure" which may be fooled by pseudo-random distributions. The nice thing about this particular task is that this search seems to require new, different proofs of the original XOR lemma, interesting in their own right. Impagliazzo's proof via the Hard-Core theorem [16] enabled him to show that pairwise independent samples achieve some nontrivial amplification (which we actually use). We give yet another proof of the lemma, based on an analysis of a simple "card guessing" game, which reveals two familiar requirements, for which the solutions (generators) are standard by now. One is the expander-walk generator of [1, 9, 19] (used for deterministic amplification) and the other is the nearly disjoint subset generator of [23] (used for fooling constant depth circuits, as well as in [25]). Our generator is simply the XOR of these two generators, together with a proof that this combination does not ruin their respective properties[1].

We conclude by putting our construction in a different context. A version of Yao's XOR-Lemma is a prototypical example of a "direct product" theorem (for the Boolean circuit model). Intuitively, such results hold for a model of computation (and bounded complexity), if whenever a function f cannot be predicted by the model with better probability than p, then the model cannot predict f on k independent instances with better probability than $p^{\Omega(k)}$. Such results exist for a variety of models besides circuits; Boolean decision trees [24], one-way communication complexity [17], 2-prover games [27] and recently also communication complexity [26]. Our result is the first derandomization of a direct product theorem. Such derandomization for other models is of interest for the similar reasons, (sometimes to be combined with "help bits" results for lower bounds) where input size blow-up is has similar consequences. Indeed, this problem was studied for the 2-prover games model, motivated by achieving better impossibility of approximation results, until Feige and Kilian [10] proved that in some formal sense derandomizing the Parallel Repetition Theorem of [27] is impossible. For which other models can direct product results be derandomized?

2 Definitions and History

In this section, we give the background needed for the paper. For simplicity, we present the results for the non-uniform Boolean circuit model of computation. Some analogs of both the historical results quoted and the new results in this paper hold for other models as well.

[1]In some contrast to the message of this paper that XOR tends to scramble things

2.1 Distibutional hardness of functions

In this sub-section, we give the standard definitions of quantitative difficulty of a function for a certain distribution of inputs. The notion of hardness for a function with a small range, with the extreme case being a Boolean function, is slightly more complicated than that for general functions, since one can guess the correct value with a reasonable probability. However, the Goldreich-Levin Theorem ([11]) gives a way to convert hardness for general functions to hardness for Boolean functions.

Definition 2.1 *Let m and ℓ be positive integers. Let $f : \{0,1\}^m \rightarrow \{0,1\}^\ell$. $S(f)$, the worst-case circuit complexity of f, is the minimum number of gates for a circuit C with $C(x) = f(x)$ for every $x \in \{0,1\}^m$.*

Let π an arbitrary distribution on $\{0,1\}^m$, and let s be an integer. Define the success $SUC_s^\pi(f)$ *to be the maximum, over all Boolean circuits C of size at most s of $Pr_\pi[C(x) = f(x)]$. Note that if f can be exactly computed in size s (i.e., $S(f) \leq s$), then $SUC_s^\pi(f) = 1$ for any distribution.*

For the (important) case $\ell = 1$ we define the advantage $ADV_s^\pi(f) = 2SUC_s^\pi(f) - 1$. *In both notations we replace the superscript π by A if π is the uniform distribution on a subset $A \subseteq \{0,1\}^m$. When π is uniform on $\{0,1\}^m$ we eliminate the superscript altogether.*

The *hard-core predicate* result of Goldreich and Levin [11] allows a hardness result for a function with a linear-size output to be converted to a hardness result for a Boolean function.

Definition 2.2 *Let $f : \{0,1\}^m \rightarrow \{0,1\}^\ell$. Define $f^{GL} : \{0,1\}^m \times \{0,1\}^\ell \rightarrow \{0,1\}$ by $f^{GL}(x,y) = < f(x), y >$, where $< \cdot, \cdot >$ denotes the inner product in $GF(2)$. For a distribution π on $\{0,1\}^m$ define π^{GL} on $\{0,1\}^m \times \{0,1\}^\ell$ be π on the first coordinate and uniform on the second, independently. The function f^{GL} is called the* Goldreich-Levin predicate *for f.*

It is easy to see that $ADV_{s+O(m)}^{\pi^{GL}}(f^{GL}) \geq SUC_s^\pi(f)$. (Use the inner product of your guess for f and y as the guess for the inner product bit. If your guess for f is correct, so is this guess. If not, the guessed bit is correct with probability 1/2.) The Goldreich-Levin hard core predicate theorem is a strong converse of this inequality. The following paraphrases the main result of [11] in our notation:

Theorem 2.3 *[11] There are functions $s' = \epsilon^{O(1)}s$, $\delta = \epsilon^{O(1)}$ so that: for every function f and distribution π, if $SUC_s^\pi(f) \leq \delta$, then $ADV_{s'}^{\pi^{GL}}(f^{GL}) \leq \epsilon$.*

Note that if $\epsilon = 1 - \gamma$, where $\gamma = o(1)$, then $\delta = (1-\gamma)^{O(1)} = 1 - O(\gamma)$. So between functions and predicates that are only weakly unpredictable the above relationship gives quite tight bounds.

2.2 Hardness versus Randomness

For this section only, we use f, g, h to denote a Boolean function on strings of all lengths, and by f_n, g_n, h_n their segments on n-bit strings. Also, let $E = DTIME(2^{O(n)})$.

The hardness versus randomness trade-off we prove is stated formally below.

Theorem 2.4 *If there is a Boolean function $f \in E$ with $S(f_n) = 2^{\Omega(n)}$, then $BPP = P$.*

Fortunately, much of the way towards this theorem was made already, and this subsection details this progress, focusing on what remains to be done.

Nisan and Wigderson [25] showed how to simulate randomized algorithms eficiently using any problem in exponential-time that is sufficiently difficult for non-uniform circuits. Their result provides a wide trade-off, but we will need only the following version.

Theorem 2.5 *[25] If there is a Boolean function $f \in E$ with $ADV_{s(n)}(f_n) = 2^{-\Omega(n)}$ for some $s(n) = 2^{\Omega(n)}$, then $P = BPP$.*

So we need a function with expoenential unpredictability. In [6], random self-reducibility of EXP-complete functions was used, to "get started", namely generate a function with very mild unpredictability, from a function hard in the worst case. The premise in Theorem 2.4 regards E, so we state below is a (somewhat tighter) variant of their results for this class. It can be proved with the same (by now standard) techniques, which we defer to the final paper. (The improvement is that our trasformation increases the input size by only a constant factor.)

Theorem 2.6 *If there is a Boolean function $f \in E$ with $S(f) = 2^{\Omega(n)}$, then there is a function $h \in E$ with $SUC_{s(n)}(h_{cn}) \leq 1 - n^{-2}$ for some constant c and some $s(n) = 2^{\Omega(n)}$.*

Yao's XOR lemma could then be used to amplify this mild hardness to the unpredictability level required in Theorem 2.4. However, this increases input size by a polynomial amount, which means that the simulation will still only be quasi-polynomial. Our main contribution will be achieving the same amplification with only a linear increase in size. Our starting point is made a bit better by the following partial amplification of [16] (proved via Theorem 2.11).

Theorem 2.7 *[16] If $h \in E$ has $SUC_{2^{\Omega(n)}}(h_n) \leq 1 - n^{-2}$, then there is a $g \in E$ with $SUC_{2^{\Omega(n)}}(g_{2n}) \leq 2/3$.*

To summarize, what remains is to, given a function with constant hardness against exponentially large circuits, "amplify" this hardness to constant another function with an exponentially small advantage against exponentially large circuits. This must be done without blowing up the input size by more than a constant factor. This is our main technical contribution, stated formally in Theorem 2.12.

2.3 Direct Product Lemmas

In order to achieve the task stated at the end of the previous section we first state the classical Yao's XOR Lemma, which accomplishes the amplification of hardness, but at the expense of increasing input size. We show its relation to other direct product lemmas, and give a definition of "derandomized" direct product lemma. We discuss known pseudo-random direct product lemmas, and state our amplification result, which is proved in the remaining sections.

Let $g : \{0,1\}^n \to \{0,1\}$ and assume $SUC_s(g) \leq 1 - \delta$. In an information theoretic analog, we could think of the value of g on a random input as being completely predictable with probability $1 - 2\delta$ and as being a random coin flip othehwise. Then the advantage of guessing the exclusive or of k independent copies of g would be $(1 - 2\delta)^k$, since if it were ever a random coin flip, we would be unable to get any advantage in predicting the XOR.

The XOR lemma is a computational manifestation of this intuition. Define $g^{\oplus(k)} : (\{0,1\}^n)^k \to \{0,1\}$ by $g(x_1, \cdots, x_k) = g(x_1) \oplus \cdots \oplus g(x_k)$.

Theorem 2.8 *[30],[20]. There are functions k, s' of n, s, ϵ, δ with $k = O(-\log \epsilon/\delta)$, $s' = s(\epsilon\delta)^{O(1)}$ and so that, for any $g : \{0,1\}^n \to \{0,1\}$ with $SUC_s(g) \leq 1 - \delta$, we have: $ADV_{s'}(g^{\oplus(k)}) \leq \epsilon$.*

Theorem 2.3 reduces the XOR lemma to the following "direct product" theorem, which requires the circuit to output *all* values of $g(x_i)$, rather than their exclusive-or.

Define $g^{(k)} : (\{0,1\}^n)^k \to \{0,1\}^k$ by $g(x_1, \cdots, x_k) = (g(x_1), \cdots, g(x_k)$. Then

Theorem 2.9 *There are $k = O(-\log \epsilon/\delta)$, $s' = (\epsilon\delta)^{O(1)}s$ so that for any $g : \{0,1\}^n \rightarrow \{0,1\}$ with $SUC_s(g) \leq 1-\delta$, we have: $SUC_{s'}(g^{(k)}) \leq \epsilon$,*

Note that the Goldreich-Levin predicate for $g^{(k)}$ is basically $g^{\oplus k/2}$, (The number of inputs is not fixed at $k/2$ but a binomial distribution with expectation $k/2$; however, if we restrict r in the Goldreich-Levin predicate to have exactly $k/2$ 1's, the bound on the advantage is almost the same.) Thus, combining Theorem 2.9 with Theorem 2.3 yields Theorem 2.8. We'll use this approach in the rest of the paper: first show that computing f on many inputs is hard and then invoke Theorem 2.3 to make the hard function Boolean.

In the next section, we give a new proof of Theorem 2.9, which generalizes to distributions on $\vec{x}$ which are not uniform and hence can be sampled from using fewer than nk random bits. This allows us to give some conditions which suffice for such a pseudo-random distribution to have direct product properties similar to the uniform distribution. We give a construction of such a pseudo-random generator which (for certain parameters of interest) uses only a linear number of bits to sample n n-bit inputs. We formalize the notion of a direct product result holding with respect to a pseudo-random generator as follows:

Definition 2.10 *Let G be a function from $\{0,1\}^m$ to $(\{0,1\}^n)^k$. G is an $(s, s', \epsilon, \delta)$ direct product generator, if for every boolean function g with $SUC_s(g) \leq 1-\delta$, we have:*

$$SUC_{s'}(g^k \circ G) \leq \epsilon$$

(Here, the distribution is over uniformly chosen inputs to G.)

Let $s, s', \epsilon, m, \delta$ and k be fixed functions of n, and let G_n be a polynomial-time computable function from $\{0,1\}^m$ to $(\{0,1\}^n)^k$. The family of functions G is called a $(s, s', \epsilon, \delta)$ direct product generator *if each G_n is a $(s(n), s'(n), \epsilon(n), \delta(n))$ direct product generator.*

In this notation, [16] proved the following, which implies Theorem 2.7.

Theorem 2.11 *[16] There is a function G from $\{0,1\}^{2n}$ to $(\{0,1\}^n)^n$ which, for any $s = s(n)$, $c \geq 1$, is a $(s, sn^{-O(1)}, O(n^{-(c-1)}), n^{-c})$ direct product generator.*

The main technical contribution of this paper is the following:

Theorem 2.12 *For any $0 < \gamma < 1$, there is a $0 < \gamma' < 1$ and a $c > 0$ so that there is a polynomial time $G : \{0,1\}^{cn} \rightarrow \{0,1\}^{n \times n}$ which is a $(2^{\gamma n}, 2^{\gamma' n}, 2^{-\gamma' n}, 1/3)$ direct product generator.*

The proof of this theorem is in Sections 3 and 4. As explained above, it establishes our hardness versus randomness trade-off Theorem 2.4. The proof will actually give the more general construction, with the above being a special case.

Theorem 2.13 *For any s, ϵ, δ there is a polynomial time $G : \{0,1\}^m \rightarrow (\{0,1\}^n)^k$, which is an $(s, s', \epsilon, \delta)$-generator with $k = O(-(\log \epsilon)/\delta)$, $s' = \sqrt{s}(\epsilon\delta/n)^{O(1)}$ and $m = O(Min\{n \log \epsilon^{-1}, n^2/\log s\}$.*

Despite the tightness of our generator for the parameters of interest, the above leaves a gap to close. We conjecture that the bounds above can be achieved for all s, ϵ, δ with $m = O(n)$.

3 The Card Guessing Game: A new proof of the direct product lemma

3.1 The Card Game

The XOR Casino offers the following game. A sequence of k cards each labelled either 0 or 1 is put on the table facing down. The cards are chosen by the casino, with the only restriction being that (by gambling laws of the state of Nevada) they must all be 1 with probability p. One card is chosen at random, and the casino turns over all cards except that one. You must guess the value of the chosen card, and recieve pay-off of \$1 if your guess is correct. What is the value of this game?

The correspondence with the direct product lemma is, intuitively, as follows. Let g be a boolean function. We want to use an algorithm A advertised as having a probability p of computing $g^k(x_1, ..x_k)$ to get an algorithm A' that predicts g on a single x. We know that A gets all k answers correct with probability p. But when A does not get the answers all correct, then the answers it gives might be anything, so we assume they are picked adversarily (by the "casino"). The value of the i'th "card" is 1 when A predicts $g(x_i)$ correctly, 0 otherwise, so we know all are 1 with probability p. We "pick" a card i by setting $x_i = x$, and "turn over" the other cards by having the values $g(x_j)$ as "advice" (if we are in a non-uniform model of computation) or by choosing x_j in a way so that we can compute $g(x_j)$ (in cryptographic settings.) A' will then incorporate our strategy for the card game to get a strategy for either believing or disbelieving the prediction of $g(x) = g(x_i)$ given by A.

We will give an almost optimal strategy for the card guessing game, and then formalize the above intuitive correspondence. Observe that if the casino, when not making all cards 1's, fixes the values as independent coin tosses, your probability is at most $p + (1-p)/2 = 1/2 + p/2$. Indeed, a slightly more careful distribution allows the casino to do a bit better, and reduce your success probability to $1/2 + p/2 - 2^{-(k-1)}$. We will show that you can get a pay-off quite close to this upper bound, and get the answer correct with probability $1/2 + p/2 - 2^{-k/3}$.

Let $\bar{a} = (a_1, a_2, \cdots, a_k) \in \{0,1\}^k$ denote the random sequence chosen by the casino, and let $I \in_R [k]$ be the picked card. Consider the following strategy S: Let t be the number of cards you see marked 0, i.e., the number of $j \neq I$ with $a_j = 0$. S guesses 1 with probability 2^{-t}, and otherwise guesses 0 or 1 uniformly at random. (The overall probability of guessing 0 is $(1-2^{-t})/2$ and that of guessing 1 is $(1+2^{-t})/2$.)

Theorem 3.1 *For any distribution on $\bar{a}$ which is equal to 1^k with probability at least p,*

$$Pr[S = a_I] \geq 1/2 + p/2 - 2^{-k/3}$$

.

Proof.

Let $\bar{a}$ be a fixed sequence of cards. Clearly, if $\bar{a} = 1^k$, then S always guesses correctly. We'll show that, for any other sequence, the conditional probability that S guesses correctly is at least $1/2 - 2^{-k/3}$.

Let T denote the number of coordinates i in which $a_i = 0$. Then with probability T/k, $a_I = 0$, and the obseved number of 0's is $t = T - 1$. In this case, S outputs 0 with probability $(1-2^{-t})/2 = 1/2 - 2^{-T}$. In the other case, which occurs with probability $1 - T/k$, $a_I = 1$, we see all T 0's and output 1 with probability $(1+2^{-T})/2 = 1/2 + 2^{-T+1}$. Thus, the overall probability that S outputs the correct answer is $1/2 + 2^{-T}((1-T/k)/2 - T/k) = 1/2 + 2^{-T}(\frac{k-3T}{2k})$. If $T \geq k/3$, this is at least $1/2$, and if $T < k/3$, it is at least $1/2 - 2^{-T} \geq 1/2 - 2^{-k/3}$.

Thus the overall probability that S gets the right answer is at least $p + (1-p)(1/2 - 2^{-k/3}) \geq 1/2 + p/2 - 2^{-k/3}$. $\square$.

We shall actually need a small variant of the above game. In the variant game, the casino picks, in addition to $\bar{a}$, a subset K of k' cards (which can depend on $\bar{a}$). The casino does not reveal K to you, but does give you a uniformly chosen element $I \in K$. You get to see the values of all cards except a_I and try to predict a_I. (Intuitively, K represents the "hard" instances of the problem.) The player can still use the strategy S described above, and a similar calculation shows:

Theorem 3.2 *For any distribution on $\bar{a}$ and K so that $\bar{a} = 1^k$ with probability at least p ,*

$$Pr[S = a_I] \geq 1/2 + p/2 - 2^{-k'/3}$$

where I is a uniformly random element from K.

3.2 (Yet Another) Proof of the XOR Lemma

Three completely different proofs of the XOR lemma are known [20, 16, 12], with the last reference containing a survey of all three. We are now ready for our new proof, after which we discuss its benefits for the purpose of derandomization.

Proof. (of Theorem 2.9) Let $q(k, \delta)$ be the probability that if k independent coins are tossed each with probability δ of heads, fewer than $\delta k/2$ heads are tossed. Then there is a constant $0 < c < 1/6$ so that $q(k,\delta) \leq 2^{-c\delta k}$. Pick $k = \frac{(-\log \epsilon + 3)}{c\delta} = O(-\frac{\log \epsilon}{\delta})$. Then $q(k, \delta) \leq \epsilon/8$, and, since $c < 1/6$, $2^{-k\delta/6} \leq \epsilon/8$. We'll show the contrapositive of the statement in Theorem 2.9: If $SUC_{s'}(g^{(k)} \geq \epsilon$, then $SUC_s(g) \geq 1 - \delta$ for some $s = s' n^{O(1)}(\epsilon^{-O(1)})$

Assume $SUC_{s'}(g^{(k)}) \geq \epsilon$, and let C be a circuit achieving this success probability. We shall use it to construct a distribution on circuits F of approximately the same size, for which $Exp_F[ADV_F^H(g)] > \epsilon/2 - 2^{-\Omega(\delta k)}$ for *any* subset H of inputs with $|H| \geq \delta 2^n$.

Then it follows that taking $t = O(n/\epsilon^2)$ samples from this distribution $F_1, ...F_t$, $Maj(F_1, ..F_t)$ (the Boolean majority function of the samples) is likely to predict f correctly on all but an at most δ fraction of inputs, contrary to the hardness assumption.

The distribution on F is merely implementing the strategy S of the previous section using $\bar{a} = 1^k \oplus C(\bar{x}) \oplus g^{(k)}(\bar{x})$.

Consider the following distribution on probabilistic circuits with one n-bit input x. $I \in [k]$ is chosen uniformly at random, and for each $J \neq I$, x_J is chosen uniformly. F sets $x_I = x$ and simulates $C(\bar{x})$. F then computes t, the number of $J \neq I$ where the J'th output of $C(\bar{x})$ differs from $g(x_J)$. (Here, we are interested in non-uniform complexity so $g(x_J)$ is given as advice; we could also modify the argument for the uniform cryptographic setting since there $x_J = f(y_J)$ and $g(x_J) = b(y_J)$ can be generated together from a common random input y_J.) F outputs the I'th output of $C(\bar{x})$ with probability 2^{-t} and otherwise outputs 0 or 1 with equal probability.

Lemma 3.3 *Let $H \subseteq \{0,1\}^n, |H| \geq \delta 2^n$. Let $q = q(\delta, k)$. Then $Exp_F[ADV_F^H(g)] > \epsilon/2 - (1.5)q - (2)2^{-\delta k/6} \geq \epsilon/16$.*

Proof. Without loss of generality, assume $|H| = \delta 2^n$. (If not, apply the result to a random $H' \subseteq H$ of the above size.) Consider the distribution on $x_1, ...x_k$ chosen by F given that $x \in H$. Any sequence $x_1, ..x_k$ with s elements from H has s values of I for which it can be chosen by F. Thus, the probabilities of such vectors are proportional to s, and since the average value of s is δk, the exact probability is $s/(\delta k)$ times the probability in the uniform distribution.

Now, there are at least a $\epsilon - q$ fraction of such tuples so that both $s \geq \delta k/2$ and $C(\bar{x}) = g^k(\bar{x})$. For each such sequence, the probability that it is chosen by F given $x \in H$ is at least 1/2 the probability that it is chosen under the uniform distribution. Therefore, the probability that F simulates C on such a tuple is at least $(\epsilon - q)/2$. Note that given that F simulates C on $\bar{x}$, I is uniformly distributed among the indices i with $x_i \in H$.

Consider the following casino strategy for the variant card-guessing game, with $k' = \delta k/2$. $I, x_1, ..x_k$ are chosen as F does, given that at least k' indices i have $x_i \in H$. The casino sets $a_i = 1$ if and only if the i'th bit of $C(\vec{x})$ is $g(x_i)$. The casino picks a subset K of k' such indices with $I \in K$ at random. ¿From the above discussion, the casino sets each $a_i = 1$ with probability at least $(\epsilon - q)/2$, and I is uniformly distributed in K. Thus, the payoff of the casino when the strategy S from Theorem 3.2 is used is at least $1/2 + (\epsilon - q)/4 - 2^{k'/3}$. Also, given that $x_1..x_k$ has at least k' elements of H, this payoff is equal to the probability that $F(x) = g(x)$, the expected success probability for F. Since the probability that $x_1, ..x_k$ does not have its quota of elements from H is at most $q/2$, the expected success probability for F is at least $1/2 + \epsilon/4 - 3q/4 - 2^{k'/3}$, and so the expected advantage is at least $\epsilon/2 - 3q/2 - 22^{k'/3}$, as claimed. □

Let $H = \{x | Prob_F(F(x) = g(x)) < 1/2 + \epsilon/32\}$. Since the expected success probability of F on H is the average of the probabilities used to define H, $Exp(Adv_F^H(g)) < \epsilon/16$. Thus, from Lemma 3.3, $|H| \leq \delta 2^n$. Standard arguments show that for t as mentioned earlier, if $F_1, ..F_t$ are chosen independently, $Maj(F_1, ...F_t)(x) = g(x)$ for all $x \notin H$ with high probability. Thus, there are $F_1, ..F_t$ so that $Maj(F_1, ..F_t)$ achieves success probability $1 - \delta$. This function will have circuit size $O(ts') = s'n^{O(1)}\epsilon^{-O(1)}$ as claimed. □

4 Direct product lemmas for pseudo-random distributions

In this section, we present some general conditions on a pseudo-random distribution on $x_1, ..x_k$ so that the proof of the direct product theorem from the last section goes through basically unchanged.

Note that the argument from the last section basically used the independence of the x_i in two ways;

- For any $H \subseteq \{0,1\}^n$, $|H| \geq \delta 2^n$, with high probability, the number of $x_i \in H$ is at least some $k' = \Omega(k)$.
- It is possible (and computationally feasible) to fix the values of all the x_J, $J \neq I$ while leaving x_I free to be any value.

Of course the last condition really does force true independence. However, if instead of x_J being *completely fixed*, x_J could be restricted to a *relatively small set of possibilities*, we could still store $g(x_J)$ for each possible x_J. This motivates the following definition:

Definition 4.1 *Let $G_n : \{0,1\}^m \to (\{0,1\}^n)^k$ be a polynomial time computable pseudo-random generator. We say G is* M-restrictable *if there is a polynomial-time computable function $h(i, x, \alpha)$, $h : [n] \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$, with the following properties. For any inputs i, x, α, let $G(h(i, x, \alpha)) = x_1, ..x_k$.*

- *For any i and uniformly chosen x,α, $h(i, x, \alpha) \in \{0,1\}^m$ is uniformly distributed.*
- *For any i, x, α, $x_i = x$.*
- *For any $i, j \neq i$, and α, there is a set $S \subseteq \{0,1\}^n$, $|S| \leq M$, so that for any value of x, $x_j \in S$.*

We also need to quantify the first, statistical use of independence:

Definition 4.2 *We say that a pseudo-random generator G producing a distribution $x_1, ..x_k$ on $(\{0,1\}^n)^k$ is (k', q, δ)-hitting if for any sets $H_1, ..H_k \subseteq \{0,1\}^n, |H_i| \geq \delta 2^n$ we have $Prob[|\{i | x_i \in H_i\}| < k'] \leq q$.*

One nice feature of the above definitions is that, if we can find two pseudo-random generators that satisfy them separately, we can merely take the bitwise XOR of the two to get a single restrictable, hitting distribution. This is formalized as follows:

Definition 4.3 *Let $x \oplus y$ denote the bit-wise xor of the two strings. Let $G(r), G'(s)$ be pseudo-random generators with outputs in $(\{0,1\}^n)^k$, $G(r) = x_1, ..x_k$ and $G'(s) = y_1, ..y_k$. Then define the pseudo-random generator $G \oplus G'$ by $(G \oplus G')(r, s) = x_1 \oplus y_1, ..., x_k \oplus y_k$.*

Lemma 4.4 *Let $G(r), G'(s)$ be polynomial-time pseudo-random generators with outputs in $(\{0,1\}^n)^k$.*

- *If G is (k', q, δ)-hitting then $G \oplus G'$ is (k', q, δ)-hitting.*
- *If G' is M-restrictable then $G \oplus G'$ is M-restrictable.*

Proof.

- Let $H_1, ..H_k \subseteq \{0,1\}^n, |H_i| \geq \delta 2^n$ be given, and fix $y_1, ..y_k$. Let $H'_i = H_i \oplus y_i$; clearly, $|H'_i| = |H_i|$. Then $x_i \oplus y_i \in H_i \iff x_i \in H'_i$, so applying the hitting condition for G to $H'_1, , ..H'_k$ gives us a bound on the hitting probability of $G \oplus G'$.
- Let h be the restricting function for G. Let $G(s) = y_1, ..y_k$. Then let $h'(i, x, \alpha \circ s) = (h(i, x \oplus y_i, \alpha), s)$. It is easy to check that h' is a restricting function for $G \oplus G'$.

We now show that these conditions are sufficient for a direct product theorem to hold for a generator:

Theorem 4.5 *Let s be a positive integer and let $G(r) : \{0,1\}^m \rightarrow (\{0,1\}^n)^k$ be a $(\rho k, q, \delta)$-hitting, M-restrictable pseudo-random generator, where $q > 2^{-\rho k/3}$ and $s > 2Mnk$. Then G is a $(s, \Omega(sq^2(n^{-O(1)})), O(q), \delta)$-direct product generator.*

Proof.

Let $\epsilon = (4(\delta/\rho) + 1)q$, We give a probabilistic construction which takes a circuit C that has success probability ϵ for $g^{(k)} \circ G$ and outputs a circuit F of size $O(|C| + kMn)$ whose expected advantage on any $H, |H| \geq \delta 2^n$ is at least q. We then take the majority of $O(n/q^2)$ such circuits.

The construction is based on the strategy for the variant card guessing game, and is similar to the previous section. We pick I from $1, ..k$ uniformly at random, and select α_0 at random. Let $x_1, ..x_k = G(h(I, x, \alpha_0))$, For each $j \neq I$, we construct (non-uniformly) a table of $g(x_j)$ for any x_j that is a possible output of $h(I, x, \alpha_0)$ (as we vary over x). There are at most M such values. Our circuit F, on input x simulates C on $h(I, x, \alpha_0)$. It then (via the tables) computes t, the number of indices $j \neq I$ where the j'th output of C differs from $g(x_j)$. F outputs the i'th output of C with probability 2^{-t}, and otherwise outputs a randomly chosen bit.

Lemma 4.6 *Let $H \subseteq \{0,1\}^n, |H| \geq \delta 2^n$. $Exp(Adv_F^H(g)) \geq q$*

Proof. Let $x \in H$ be uniformly chosen. Let u be the number of $x_i \in H$. As in the previous section, the probability of a sequence $x_1, ..x_k$ given that $x \in H$ is $u/(\delta k)$ times its probability of being output by $G(r)$. The probability, for a random $G(R)$, that $u \geq \rho k$ and that $C(r) = g^k(G(r))$ is at least $\epsilon - q = 4\delta/\rho q$ from the hitting property of G and the success probability of C . Thus, the corresponding probability that $u \geq \rho k$ and C succeeds, for $x_1, ..x_k$ constructed by F on a random $x \in H$, is at least $(\epsilon - q)\rho/\delta = 4q$, since each such sequence has at least ρ/δ of its probability under $G(r)$. Also, $r = h(I, x, \alpha_0)$ is independent of I so given r (and hence $x_1, ..x_k$), I is a uniformly distributed index so that $x_I \in H$.

So again, the expected success of F on $x \in H$ given that $u \geq \rho k$ is the same as the expected advantage of S on the variant game with $k' = \rho k$ when the casino sets $a_j = 1$ iff the j'th output of C is $g(x_i)$, and picks K to be a random set of k' positions i with $x_i \in H$. ¿From the above, the probability p that all the a_i are 1 is at least $4q$. So by Theorem 3.2, the expected success probability of F given $u \geq k'$ is at least $1/2 + p/2 - 2^{-\rho k/3} \geq 1/2 + 2q - 2^{\rho k/3} \geq 1/2 + q$. Then, since $u \geq k'$ with probability at least $1 - q$, the overall expected success probability of F is at least $(1 - q)(1/2 + q) \geq 1/2 + q/2$, which yields the claimed advantage. □

Thus, there is at most a δ fraction of x where F has expected probability $< 1/2+q/2$, and taking the majority of $O(n/q^2)$ circuits F gives a circuit that predicts $g(x)$ on all x not in this fraction. Each F has size approximately $O(|C| + kM)$, so the combined circuit has size $O((|C| + kMn)n/q^2)$. Thus, if $|C| \leq s'$, there is a circuit of size $\leq s$ that has success probability at least $1 - \delta$, a contradiction. □

5 Examples of hitting and restrictable generators

In this last section, we reduced the question of constructing a direct product generator to that of (separately) constructing hitting and restrictable generators. Here, we present a well-known example of each type, and combine them. This completes the proofs of Theorem 2.12 and Theorem 2.13.

5.1 Expander Walks: a hitting generator

Definition 5.1 *Fix an explicit expander graph G on vertex set $\{0,1\}^n$, of constant degree (say 16) and small second eigenvalue (say < 8) [22]. Let the generator $EW : \{0,1\}^n \times [16]^{k-1} \to (\{0,1\}^n)^k$ be defined by $EW(v; \bar{d}) = (v_1, v_2, \cdots, v_k)$ where $v_1 = v$ and v_{i+1} is the d_ith neighbour of v_i in G.*

Note that this generator uses a seed of length $m = n + O(k)$. It was first considered in [1]. The amplification property of this generator, formalized below, asserts that with exponentially high probability, many of these k pseudo random points will satisfy arbitrary large events.

Theorem 5.2 *[9, 19] For every $k \leq n$, the above generator is $(k/10, 2^{-k/10}, 1/3)$ hitting.*

5.2 Nearly Disjoint Subsets: a restrictable generator

This generator will in general use more than $O(n)$ bits, but for the parameters of interest in Theorem 2.12 uses only $O(n)$ bits. The number of bits given in Theorem 2.13 for general circuit size will follow from the seed length here, and any improvement should circumvent this construction. We mention that this is impossible to do by simply building more efficient restrictable generators! [18] use entropy arguments to prove that the bounds below are tight for any construction.

Let M be given, let $\gamma = (\log M)/n$, i.e., $M = 2^{\gamma n}$, and set $m = 2n/\gamma$. In this section, we construct an M-restrictable generator using $O(m)$ bits. Unfortunately, $O(m)$ bits is necessary for such a generator. Thus, the direct product generator we construct is only useful when $s > M > 2^{O(\sqrt{(n)})}$; otherwise m is larger than the number of bits in $O(\log s)$ independent instances. However, if we are dealing with exponentially hard functions, $M = 2^{\Omega(n)}$ and so $m = O(n)$.

Definition 5.3 *Let $\Sigma = \{S_1, ...S_k\}$ be a family of subsets of $[m]$ of size n. We say Σ is γ-disjoint if $|S_i \cap S_j| \leq \gamma n$ for each $i \neq j$. Let $r \in \{0,1\}^m$ and $S = \{s_1 < s_2 < ...s_n\} \subseteq [m]$. Then let the restriction of r to S, $r|_S$, be the n-bit string $r|_S = r_{s_1} r_{s_2} .. r_{s_n}$. Then, for a γ-disjoint Σ, $ND^\Sigma : \{0,1\}^m \times \to (\{0,1\}^n)^k$ is defined by: $ND^\Sigma(r) = r|_{S_1}, r|_{S_2}, \cdots, r|_{S_k}$.*

Lemma 5.4 *There is a polynomial time probabilistic algorithm that for any $\gamma > 0, n, k = O(n)$ and $m = 2n/\gamma$, produces, with probability $1 - 2^{-\Omega(n)}$, a γ-disjoint family of k subsets of $[m]$ each of size n, and uses $O(m)$ bits.*

Proof. Let $p = \binom{m}{n} \leq 2^m$ and encode (efficiently and 1-1) the n-subsets of m by the integers modulo p. Pick n numbers in Z_p so that they are individually uniform and pairwise independent [9],[8] and let the S_i's be the sets encoded by these numbers. The probability that, for two independent sets S and S' of size n, their intersection is more than twice the expected size $m(n/m)^2 = n^2/m = \gamma n/2$ is exponentially small. Thus, since the S_i's are pairwise independent, this exponetially small amount times nk bounds the probability that any $|S_i \cap S_j| > \gamma n$ for $i \neq j$.) $\square$

Lemma 5.5 *If Σ is γ-disjoint, then ND^Σ is $M = 2^{\gamma n}$-restrictable.*

Proof. For $i \in [n], x \in \{0,1\}^n, \alpha \in \{0,1\}^{m-n}$, let $h(i, x, \alpha)$ be the string $r \in \{0,1\}^m$ with $r_{S_i} = x$ and $r_{\overline{S_i}} = \alpha$. Fix i , $j \neq i$, and α. Since $|S_j \cap S_i| \leq \gamma n$, and since the output of $ND(h(i, x, \alpha))$ is determined in all bits not in S_i, there are only $2^{\gamma n}$ possible values for x_j, found by varying all bits in $S_j \cap S_i$.

5.3 The Direct Product (and XOR) Generator XG

The generator we use to "fool" the XOR lemma is the simplest combination of the two generators above - we take the componentwise exclusive-or of their outputs. We describe here only the choice of parameters required by Theorem 2.12.

Definition 5.6 *Let n be a positive integer, and let $1/30 > \gamma > 0$. Let $k = n$ in EW, and let $\gamma' = \gamma/4$, and let $m = 2n/\gamma'$. With the parameters above, we define the XOR generator $XG(r; r'; v; \bar{d})$ as follows: Use r to select Σ, a γ'-disjoint family of subsets of $[m]$. Then $XG[r; r'; v; \bar{d}) = ND^\Sigma(r') \oplus EW(v, \bar{d})$.*

We ignore the possibility that Σ is not γ'-disjoint, since this happens only with exponentially small probability.

We can now prove Theorem 2.12:

Proof. We need to construct a $(2^{\gamma n}, 2^{\gamma' n}, 2^{-\gamma' n}, 1/3)$-direct product generator whose input length is $O(n)$. We claim that XG is the desired generator. First note that $|r| = O(n)$, $|r'| = m = O(n)$, $|v| = n$, and $|\bar{d}| = O(k) = O(n)$. So the input to G is $O(n)$ bits.

By lemma 5.5 , Theorem 5.2, and Lemma 4.4, XG is $M = 2^{\gamma' n}$-restrictible, and is $(n/10, 2^{-n/10}, 1/3)$-hitting. Then by Theorem 4.5, XG is a $(s, O((s - Mnk)q^2/n), O(q), 1/3)$-direct product generator for any $q > 2^{-k/30}$. In particular, for $s = 2^{\gamma n}$, $q = 2^{-\gamma' n}$, $XG2$ is a $(2^{\gamma n}, 2^{(\gamma/2)n}/n^{O(1)}, O(2^{-\gamma' n}), 1/3)$-direct product generator.

References

[1] M. Ajtai, J. Komlos and E. Szemeredi, "Deterministic simulation in LOGSPACE", Proc. of *19th ACM STOC*, 132-140, 1987.

[2] N. Alon, "Eigenvalues and Expanders", *Combinatorica*, 6, pp. 83-96, 1986.

[3] N. Alon and J. Spencer , *The Probabilistic Method*, Wiley-Interscience Publication, 1992.

[4] A. Andreev, A. Clementi and J. Rolim, "Hitting Sets Derandomize BPP", in *XXIII International Colloquium on Algorithms, Logic and Programming (ICALP'96)*, 1996.

[5] A. Andreev, A. Clementi, and J. Rolim, "Hitting Properties of Hard Boolean Operators and its Consequences on BPP", manuscript, 1996.

[6] L. Babai, L. Fortnow, N. Nisan and A. Wigderson, "BPP has Subexponential Time Simulations unless EXPTIME has Publishable Proofs", *Complexity Theory*, Vol 3, pp. 307–318, 1993.

[7] M. Blum and S. Micali. "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. Comput.*, Vol. 13, pages 850–864, 1984.

[8] B. Chor, and O. Goldreich, "On the Power of Two-Point Based Sampling", *J. of Complexity*, 5, 96-106, 1989.

[9] A. Cohen and A. Wigderson, "Dispensers, Deterministic Amplification, and Weak Random Sources", *30th FOCS*, pp. 14–19, 1989.

[10] U. Feige and J. Kilian,

[11] O. Goldreich and L.A. Levin. "A Hard-Core Predicate for all One-Way Functions", in *ACM Symp. on Theory of Computing*, pp. 25–32, 1989.

[12] [GNW] O. Goldreich, N. Nisan and A. Wigderson. "On Yao's XOR-Lemma", available via www at ECCC TR95-050, 1995.

[13] Goldreich, O., Impagliazzo, R., Levin, L., Vanketesan, R., Zuckerman, D., "Security Preserving Amplification of Harness", 31^{st} *FOCS*, pp. 318-326, 1990.

[14] S. Goldwasser and S. Micali. "Probabilistic Encryption", *JCSS*, Vol. 28, pages 270–299, 1984.

[15] J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby, "Construction of Pseudorandom Generator from any One-Way Function", to appear in *SICOMP*. (See preliminary versions by Impagliazzo et. al. in *21st STOC* and Hastad in *22nd STOC*.)

[16] R. Impagliazzo, "Hard-core Distributions for Somewhat Hard Problems", in *36th FOCS*, pages 538–545, 1995.

[17] R. Impagliazzo, R. Raz. A. Wigderson, "A Direct Product Theorem", *Proc. of the 9th Structures in Complexity conference*, pp. 88–96, 1994.

[18] R. Impagliazzo, A. Wigderson, *An Information Theoretic Analog of the Inclusion–Exclusion Lower Bound*, Manuscipt 1996.

[19] R. Impagliazzo and D. Zuckerman, "How to Recycle Random Bits", *30th FOCS*, pp. 248-253, 1989.

[20] L.A. Levin, "One-Way Functions and Pseudorandom Generators", *Combinatorica*, Vol. 7, No. 4, pp. 357–363, 1987.

[21] R. Lipton, "New directions in testing", In J. Fegenbaum and M. Merritt, editors, *Distributed Computing and Cryptography*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science Volume 2, pp. 191-202. American Mathematical Society, 1991.

[22] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs", *Combinatorica*, 8(3):261–277, 1988.

[23] N. Nisan, "Pseudo-random bits for constant depth circuits", *Combinatorica 11* (1), pp. 63-70, 1991.

[24] N. Nisan, S. Rudich, and M. Saks. "Products and Help Bits in Decision Trees", in *35th FOCS*, pages 318–329, 1994.

[25] N. Nisan, and A. Wigderson, "Hardness vs Randomness", *J. Comput. System Sci.* 49, 149-167, 1994

[26] I. Parnafes, R. Raz, A. Wigderson, *Direct Product Results, the GCD Problem, and New Models of Communication*, Manuscript 1996.

[27] R. Raz, "A Parallel Repetition Theorem", to appear in *SIAM Journal on Computing.* Preliminary version in *27th STOC*, pp. 447-456, 1995.

[28] S. Rudich, "NP-Natural Proofs", In preparation.

[29] M. Sipser, "A complexity-theoretic approach to randomness".

[30] A.C. Yao, "Theory and Application of Trapdoor Functions", in *23st FOCS*, pages 80–91, 1982.