Weak derandomization of weak algorithms: explicit versions of Yao's lemma

Ronen Shaltiel
Department of Computer Science
University of Haifa
Haifa, Israel
ronen@cs.haifa.ac.il

***Abstract*—A simple averaging argument shows that given a randomized algorithm A and a function f such that for every input x, $\Pr[A(x) = f(x)] \geq 1-\rho$ (where the probability is over the coin tosses of A), there exists a *nonuniform* deterministic algorithm B "of roughly the same complexity" such that $\Pr[B(x) = f(x)] \geq 1 - \rho$ (where the probability is over a uniformly chosen input x). This implication is often referred to as "the easy direction of Yao's lemma" and can be thought of as "weak derandomization" in the sense that B is deterministic but only succeeds on most inputs. The implication follows as there exists a fixed value r' for the random coins of A such that "hardwiring r' into A" produces a deterministic algorithm B. However, this argument does not give a way to *explicitly construct* B.**

In this paper we consider the task of proving *uniform versions* of the implication above. That is, how to *explicitly construct* a deterministic algorithm B when given a randomized algorithm A. We prove such derandomization results for several classes of randomized algorithms. These include: randomized communication protocols, randomized decision trees (here we improve a previous result by Zimand), randomized streaming algorithms and randomized algorithms computed by polynomial size constant depth circuits.

Our proof uses an approach suggested by Goldreich and Wigderson and "extracts randomness from the input". We show that specialized (seedless) extractors can produce randomness that is in some sense not correlated with the input. Our analysis can be applied to *any* class of randomized algorithms as long as one can explicitly construct the appropriate extractor. Some of our derandomization results follow by constructing a new notion of seedless extractors that we call "extractors for recognizable distributions" which may be of independent interest.

***Keywords*-Randomized algorithms, Derandomization, Randomness extractors;**

I. Introduction

A. Background

Randomized algorithms and derandomization: Randomized algorithms are algorithms that get an additional input which is a sequence of independent coin tosses and are allowed to err with small probability. For some choices of computational resources it is known that randomized algorithms can be significantly more efficient than deterministic ones. For example, when measuring communication complexity (that is the amount of communication exchanged by two parties each holding "half of the input") there are tasks that require a linear amount of communication by deterministic algorithms whereas randomized algorithms can use a logarithmic amount of communication. When the complexity measure is running time, a longstanding open problem asks whether $BPP = P$ (namely, can any randomized polynomial time algorithm be simulated by a polynomial time deterministic algorithm). A long line of research is devoted to studying this problem (see e.g. [27], [19], [15] for survey articles). Some highlights of this research are "hardness versus randomness tradeoffs" showing that $BPP = P$ assuming certain circuit lower bounds [17] (see also [5], [46], [32], [4], [40], [39], [43]), and that the statement $BPP = P$ entails certain circuit lower bounds that seem hard to prove using current techniques [20].

Deterministic algorithms that do well on a random input: A simple averaging argument (that is due to Yao [48]) shows that given a randomized algorithm A that computes a function f correctly with high probability over its random coins, there exists a deterministic algorithm B that computes f correctly with high probability over a random input. More precisely, given functions $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, $f : \{0,1\}^n \to \{0,1\}$ and a number $0 < \rho < 1/2$ we say that A *computes* f with *success* $1 - \rho$ if

$$\forall x \in \{0,1\}^n : \Pr_{R \leftarrow \{0,1\}^m}[A(x,R) = f(x)] \geq 1 - \rho \quad (1)$$

An averaging argument gives that under this assumption there exists a string $r' \in \{0,1\}^m$ such that setting $B(x) = A(x,r')$ gives that the *deterministic algorithm* B satisfies:

$$\Pr_{X \leftarrow \{0,1\}^n}[B(X) = f(X)] \geq 1 - \rho \quad (2)$$

Yao's lemma: The aforementioned implication is often referred to as the "easy direction" of Yao's lemma [48]. In this paper we are only interested in the "easy direction" and we refer to it as "Yao's lemma".[1] Yao's lemma can

[1]The "hard direction" which follows using von-Neumann's min-max theorem shows that if (2) holds for any probability distribution over the inputs then (1) holds. We remark that the proof of the "easy direction" gives a stronger implication than stated above in which (2) holds for any probability distribution over the inputs.

be viewed as a "weak derandomization" of randomized algorithms in the sense that it converts a randomized algorithm A into a deterministic algorithm B. For many complexity measures (e.g. communication complexity, circuit complexity) the deterministic algorithm B has essentially the same complexity as A. However, the argument also has two obvious weaknesses:

- The deterministic algorithm B doesn't necessarily succeed on *all* inputs and is only guaranteed to succeed with high probability on a random input.
- The averaging argument only shows the *existence* of the deterministic algorithm B but it does not give an *explicit* way to construct it.

To demonstrate the second weakness, note that if A is computable by a polynomial time Turing machine we cannot deduce that there exists a polynomial time Turing machine B that satisfies (2). This is because the averaging argument only shows the *existence* of a string r' and doesn't give an explicit way to find it. We can however deduce that there is a polynomial size *circuit* B that satisfies (2) because we can hardwire the constant string r' into the circuit A to create the circuit $B(x) = A(x, r')$.

Summing up this discussion we want to distinguish between the *complexity* of an algorithm (which is the amount of resources used by the algorithm) and the *uniformity* of an algorithm (namely, whether it can be *explicitly constructed* in uniform polynomial time). Using this terminology, when given a randomized algorithm A that satisfies (1), Yao's lemma gives a deterministic algorithm B that satisfies (2) such that B has essentially the same complexity as A but the lemma does not guarantee that B is explicitly constructible.

We will be interested in giving transformations that given a randomized algorithm A that succeeds with high probability on every input, *explicitly construct* a deterministic algorithm B (with roughly the same complexity as A) that succeeds with high probability on a random input. We will refer to this goal as achieving "an explicit version of Yao's Lemma" or "explicit weak derandomization".

Adelman's theorem [1] states that $BPP \subseteq P/poly$, namely that any uniform polynomial time randomized algorithm can be simulated by a polynomial size circuit that succeeds on *all inputs*. The source of non-uniformity in Adelman's Theorem is that Yao's Lemma does not provide an explicitly constructible deterministic algorithm.[2] In particular, giving an explicit version of Yao's Lemma that applies to all polynomial time randomized algorithms and every choice of ρ implies $BPP = P$.

B. Explicit versions of Yao's lemma for weak algorithms

We want to prove unconditional results and therefore we limit our attention to weak classes of randomized algorithms. In this paper we prove explicit versions of Yao's lemma for communication protocols and decision trees. (These results require that the number of random coins tossed by the algorithm is smaller than the input length). In addition we prove explicit versions of Yao's Lemma for streaming algorithms and algorithms implemented by constant depth circuits (without a restriction on the number of coin tosses). We also give *conditional* results that give explicit versions of Yao's lemma for *general* polynomial time algorithms assuming that certain hard functions exists. These conditional results are incomparable to known "hardness versus randomness tradeoffs" in the sense that they use different assumptions and provide derandomization that only succeeds on most inputs. We survey our results below.

1) Communication protocols: Communication complexity was first defined by Yao [47] (see the book [26] for a comprehensive treatment). In this setup we think of an input $x \in \{0,1\}^n$ as a pair of inputs $x = (x_1, x_2)$ where $x_1, x_2 \in \{0,1\}^{n/2}$. There are two parties P_1, P_2 where P_i receives x_i and the two parties want to compute a function $f(x_1, x_2)$ by exchanging few bits of communication. Both deterministic and randomized communication protocols are considered and the complexity of a communication protocol is the number of bits exchanged on the worst input x.

Yao's lemma gives "weak derandomization" for communication protocols and as noted earlier we cannot expect a "strong derandomization" of the form of Adelman's theorem in the setting of communication complexity.[3] We prove an explicit version of Yao's lemma for communication protocols. Namely, when given a randomized (public coin) communication protocol A we show how to explicitly construct a deterministic protocol B with related complexity that succeeds on "most inputs". In order to state our results we consider the notion of *explicitly constructible* (or in other words *uniform*) communication protocols.

Loosely speaking, a protocol is *explicitly constructible* if whenever a party P_i needs to send a bit in the communication protocol, this bit can be computed in *polynomial time* as a function of the party's input x_i and the previously

[2] More precisely, the proof of Adelman's theorem works in two steps:

- *Amplification:* Given a randomized algorithm $A(x, r)$ which succeeds in computing a function f on all inputs $x \in \{0,1\}^n$ with probability $2/3$ the algorithm is amplified (by running it several times with independent random coins) so that it succeeds with probability $1 - \rho$ for $\rho = 2^{-2n}$.
- *Yao's lemma* Applying Yao's Lemma, there exists a deterministic algorithm $B(x)$ such that $\Pr_{X \leftarrow \{0,1\}^n}[B(X) \neq f(X)] < \rho = 2^{-2n} < 2^{-n}$. Note that any positive probability in this probability space is at least 2^{-n} and therefore the probability above is zero. This implies that B succeeds on *all inputs*.

[3] The reason that the aforementioned proof of Adelman's Theorem does not go through is that the *amplification step* does not preserve the complexity of communication protocols. More precisely, amplifying from $\rho = 1/3$ to $\rho = 2^{-2n}$ requires multiplying the complexity by $\log(1/\rho) \geq n$ which gives a trivial communication protocol (as any function can be computed by a deterministic communication protocol of complexity n). We remark that less ambitious amplification to larger values of ρ (e.g. $\rho = 1/n$) is sometimes "affordable" for communication protocols.

communicated bits. In the case of randomized (public coin) communication protocols the polynomial time machine also gets the string of random coin tosses as an additional input. Note that as this is an asymptotic notion, we need to consider families $A = \{A_n\}$ of communication protocols (where A_n takes inputs of length n) for this to make sense. A precise formal definition appears in the full version.

The issue of whether protocols that are explicitly constructible is rarely addressed in communication complexity. This is mostly because most of the research in communication complexity is concerned with *lower bounds* and can handle protocols that are not explicitly constructible. However, we believe that when providing *upper bounds* by designing communication protocols we should prefer communication protocols that are *explicitly constructible*. The majority of protocols that appear in the literature are explicitly constructible. However, there are some exceptions.[4]

We show that given a randomized communication protocol with communication complexity q and m coins we can *explicitly construct* a deterministic communication protocol of communication complexity $O(q + m)$ that simulates A on most inputs.

Theorem 1: (*Explicit version of Yao's lemma for communication protocols*): There exists a constant $\beta > 0$ such that the following holds: Let $A = \{A_n\}$ be an explicitly constructible family of randomized communication protocols with complexity $q(n)$ and $m(n) \leq \beta n - q(n)$ coins such that for every n, A_n computes a function $f_n : \{0,1\}^n \to \{0,1\}$ with success $1 - \rho(n) \geq 2/3$. Then, there is an explicitly constructible family $B = \{B_n\}$ of deterministic communication protocols with complexity $O(q(n) + m(n))$ such that for every n, $\Pr_{X \leftarrow \{0,1\}^n}[B_n(X) = f_n(X)] \geq 1 - \rho(n)$.

Under the weaker assumption that A succeeds with probability $1 - \rho(n)$ when *both* the input and the random coins are chosen uniformly at random, namely that for every n,

$$\Pr_{X \leftarrow \{0,1\}^n, R \leftarrow \{0,1\}^{m(n)}}[A_n(X, R) = f_n(X)] \geq 1 - \rho(n)$$

we obtain that B succeeds with probability $1 - 3\rho(n) - 2^{-10m(n)}$ on a uniformly chosen input. This also applies for all our results on other classes of weak algorithms that are described next.

2) Decision trees: A decision tree of complexity q is a procedure that is allowed to adaptively make q queries of the form "what is the i'th bit of the input x" and outputs a value at the end. The reader is referred to [6] for a survey article on decision trees. We want to discuss explicit versions of Yao's lemma and consider *explicitly constructible* (or *uniform*) decision trees. We say that a (family of) decision trees is *explicitly constructible* if there is a polynomial time Turing machine which implements the decision tree. Namely when given the answers to the queries made so far the machine produces the index i of the bit to be queried next, and runs in time polynomial in $|i| = \log n$ and $q(n)$ (where $q(n)$ is the depth of the tree on inputs of length n). For a randomized decision tree with $m(n)$ coins the machine also receives a string $r \in \{0,1\}^{m(n)}$. A precise formal definition appears in the full version. In particular, sublinear time randomized algorithms that run in time $q(n) < n$ and have random access to the input, are captured by randomized decision trees with complexity $q(n)$ and $m(n) \leq q(n)$ random coins.

Zimand [50] shows how to achieve weak derandomization of sublinear time algorithms. In our terminology Zimand's result can be seen as an explicit version of Yao's Lemma for decision trees. Namely, given a randomized decision tree A with complexity $q \ll n$ and $m = q$ coins, one can explicitly construct a deterministic decision tree B that simulates A correctly on most inputs. A precise formulation using our notation follows:[5]

Theorem 2: (*Explicit version of Yao's lemma for decision trees [50]*): There exists a constant $\alpha > 0$ such that the following hold: If there is an explicitly constructible family $A = \{A_n\}$ of randomized decision trees with complexity $q(n) \leq n^\alpha$ and $m(n) = q(n)$ coins such that for every n, A_n computes a function $f_n : \{0,1\}^n \to \{0,1\}$ with success $1 - \rho(n) \geq 2/3$. Then, there is an explicitly constructible family $B = \{B_n\}$ of deterministic decision trees with complexity $q(n)^{O(1)}$ such that for every n, $\Pr_{X \leftarrow \{0,1\}^n}[B_n(X) = f_n(X)] \geq 1 - \rho(n)$.

We remark that Zimand's theorem (as well as our results that appear next) applies in a more general setting for decision trees that approximate functions or solve search problems.[6] Our techniques provide an alternative proof of Zimand's result. Our proof gives a quantitative improvement over Zimand's result in the sense that we get a deterministic decision tree with complexity that is *linear* in $q(n)$ whereas Zimand gives a deterministic decision tree of complexity

[4]To name one example, a classical result by Newman [28] states that any randomized communication protocol can be simulated by one that has $m = O(\log n)$ coins (this is used to show the equivalence between public coin and private coin models of randomized communication complexity). It is interesting to note that the proof uses a probabilistic argument and does not give an explicitly constructible protocol. It is open whether one can explicitly construct such a communication protocol. There are also other examples in the literature where the computation performed by the parties takes exponential time.

[5]Zimand states a stronger quantitative result in which B succeeds with probability $1 - 2^{-\Omega(q(n) \log q(n))}$ for $\rho = 1/3$. The two formulations are equivalent as if one wants to improve the success probability from say $1 - \rho = 2/3$ to $1 - 2^{-q(n) \log q(n)}$ it is possible to first amplify the success probability of A to that value. This increases the complexity of A by at most a polynomial. One can then apply the weaker statement of Theorem 2 on the amplified algorithm to get the stronger statement in Zimand's paper.

[6]For decision trees that *compute* functions it is known that there is at most a polynomial gap between deterministic complexity and randomized complexity [29]. (Note that this is very different than communication protocols in which there may be exponential gaps). Theorem 2 (and all our results) also apply to algorithms that "approximate functions" or solve "search problems". We elaborate on these issues in the full version. We stress that in this setup there are exponential gaps between deterministic and randomized decision trees. Note for example that randomized decision trees can approximate the number of ones in the input with logarithmic complexity whereas deterministic decision trees require linear complexity.

that is a large polynomial in $q(n)$. (In his paper Zimand estimates that the polynomial in his proof is $q(n)^{24}$). The parameters that come up in the formal statement below are identical to those in Theorem 1.

Theorem 3: (*Improved explicit version of Yao's lemma for decision trees*): There exists a constant $\beta > 0$ such that the following holds: If there is an explicitly constructible family $A = \{A_n\}$ of randomized decision trees with complexity $q(n)$ and $m(n) \leq \beta n - q(n)$ coins such that for every n, A_n computes a function $f_n : \{0,1\}^n \to \{0,1\}$ with success $1 - \rho(n) \geq 2/3$. Then, there is an explicitly constructible family $B = \{B_n\}$ of deterministic decision trees with complexity $O(q(n) + m(n))$ such that for every n, $\Pr_{X \leftarrow \{0,1\}^n}[B_n(X) = f_n(X)] \geq 1 - \rho(n)$.

3) Streaming algorithms: Streaming algorithms (see e.g. [3], [10]) are algorithms which read the input in "one pass" using "small" space. Here the complexity measure is the space used. Our techniques give an explicit version of Yao's lemma for streaming algorithms with the same parameters as Theorems 1,3.

One can consider two variants of randomized streaming algorithms. The first variant allows the randomized algorithm to store its random coins on a separate tape where this tape does not "count" when we measure the space of the algorithm. This strong variant of streaming algorithms is the one that we handle in the result above. A weaker and more common variant of randomized streaming algorithms is that of an algorithm that "tosses coins on the fly" while reading the input. Such an algorithm needs to use its internal memory if it wants to store the outcomes of the random coins and is charged for doing so. When considering the second variant we can handle randomized algorithms that toss *many* (say polynomial number of) bits. This is because we can transform such a streaming algorithm into a randomized streaming algorithm with a polylogarithmic number of random coins using pseudorandom generators for small space algorithms [30], [33], [16], [36] and then work with the latter algorithm. This approach produces deterministic streaming algorithms where the space increases by an additive polylogarithmic factor. The precise statement of our results on streaming algorithms appear in the full version.

We remark that in the setup of streaming algorithms the aforementioned pseudorandom generators do not yield a derandomization that succeeds on all inputs. This is because the standard approach of decreasing the number of random coins and then enumerating all coin values yields an algorithm that makes *many passes* when reading the input.

4) Constant depth algorithms: We consider algorithms that are computable by uniform families of polynomial size constant depth circuits. This class is called uniform-AC^0.[7] We refer to such a family as a *deterministic uniform* AC^0 *algorithm*. There is also a corresponding notion of a randomized algorithms $A(x, r)$. Loosely speaking, it is required that $|r|$ is polynomial in $|x|$ and that the function $A(x, r)$ is in uniform-AC^0. More precisely, a *uniform randomized* AC^0 *algorithm* is a family of functions $A = \{A_n\}$ such that $A_n : \{0,1\}^n \times \{0,1\}^{m=\text{poly}(n)} \to \{0,1\}$ and the family A is in uniform-AC^0. (This class is sometimes referred to as $BPAC^0$).

Full derandomization a-la Adelman's theorem applies in this setup (using the fact that approximate majority is in AC^0 [2]) and gives that (nonuniform) randomized AC^0 and (nonuniform) deterministic AC^0 coincide. For uniform algorithms there are beautiful constructions of pseudorandom generators against AC^0 [32]. These constructions are based on the hardness on average of the parity function against AC^0 [14]. Applying these generators gives that every uniform randomized AC^0 algorithm can be simulated by a (deterministic) uniform family of constant depth circuits that has *quasi-polynomial* size [32], [24]. In other words, these results give uniform *strong* derandomization (a simulation that succeeds on *all* inputs) but the resulting algorithm has quasi-polynomial size (that is size $n^{O(\log n)}$) rather than polynomial size.

We prove an explicit version of Yao's lemma for AC^0 algorithms. This gives the following incomparable result: For every uniform randomized AC^0 algorithm there is a uniform deterministic AC^0 algorithm that succeeds on most inputs.

Theorem 4 (Explicit version of Yao's lemma for AC^0): Let $A = \{A_n\}$ be a randomized uniform AC^0 algorithm such that for every n, A_n computes a function $f_n : \{0,1\}^n \to \{0,1\}$ with success $1 - \rho(n) \geq 2/3$. Then, for every constant v there is a deterministic uniform AC^0 algorithm $B = \{B_n\}$ such that for every n, $\Pr_{X \leftarrow \{0,1\}^n}[B_n(X) = f_n(X)] \geq 1 - \rho(n) - n^{-v}$.

Note that in contrast to our previous results, Theorem 4 does not assume that the number of random coins of A is smaller than the length of the input. This is because by the aforementioned pseudorandom generators we can assume w.l.o.g. that a uniform randomized algorithm uses polylog(n) random coins, and we lose only a small quantity in the success probability. This small loss in the success probability is the reason for the factor n^{-v} above. Note that by amplifying the success probability we can assume without loss of generality that $\rho(n) \leq 2^{-2n}$. Thus, if one could remove the additive factor of n^{-v} one would get

[7] Uniformity means that there is a uniform machine that on input n constructs the appropriate circuit in the family. Different variants of this notion are obtained when considering machines from different complexity classes. In this paper we use the standard notion that the machine runs in space logarithmic in n and the size of the circuit. However, our results apply also to other notions.

a deterministic algorithm that succeeds on all but a 2^{-2n} fraction of inputs of length n, which in turn implies that the deterministic algorithm succeeds on all inputs. We stress however that we cannot expect to prove such a result using the approach of this paper.

5) Conditional weak derandomization for general polynomial time algorithms: We now consider the class of *general* polynomial time randomized algorithms. Here we give an explicit version of Yao's lemma under a hardness assumption. This result follows by imitating the proof of Theorem 4 and replacing the parity function (which is hard for AC^0) with a function that is assumed to be hard on average for polynomial size circuits. Loosely speaking, a useful property of the parity function is that it is very hard on average for "weak" circuits, yet can be computed precisely by circuits with slightly larger resources. Generalizing this to our setup gives the assumption that for every constant c there are functions that are hard on average for circuits of size n^c and yet are computable in time n^d for some constant $d > c$.

Theorem 5: (*Conditional explicit version of Yao's lemma for polynomial time algorithms*): Assume that there exists a constant $\gamma > 0$ such that for every constant c there exists a constant d and a family of functions $h = \{h_n\}$ where $h_n : \{0,1\}^n \to \{0,1\}$, such that h is computable in time n^d and for every sufficiently large n and every circuit C of size n^c, $\Pr_{X \leftarrow \{0,1\}^n}[C(X) = h_n(X)] \leq 1/2 + 2^{-n^\gamma}$. Then there exists a constant $\eta > 0$ such that for every language $L \in BPP$ there is a polynomial time deterministic algorithm B that for every n, algorithm B decides the language L correctly on a $1 - 2^{-n^\eta}$ fraction of inputs of length n.

It is natural to compare Theorem 5 to hardness versus randomness tradeoffs [32], [17]. The latter give a stronger conclusion of a deterministic algorithm that succeeds on all inputs. On the other hand, the assumption of Theorem 5 requires lower bounds against circuits of polynomial size. This is usually referred to as "the low-end of hardness assumptions" in the literature on hardness versus randomness tradeoffs and these tradeoffs produce deterministic algorithms that run in subexponential time (and not polynomial time). Another difference is that Theorem 5 requires that the hard function is computable in polynomial time whereas hardness versus randomness tradeoffs allow the hard function to be computable in exponential time.

Some previous work on hardness versus randomness tradeoffs [18], [42] considered "uniform tradeoffs" in which the deterministic algorithm is guaranteed to succeed with high probability on every samplable distribution. This notion of "weak derandomization" is stronger than the one in this paper which only considers the uniform distribution. We point out that similar to the aforementioned "non-uniform" hardness versus randomness tradeoffs, the tradeoffs in [18], [42] only produce subexponential time deterministic algorithms when starting from a "low-end hardness assumption". We remark that the "uniform tradeoffs" only need to assume hardness for uniform randomized algorithms rather than circuits.

Theorem 5 can be compared to a result of [13] that also gets weak derandomization under the assumptions that there exist functions that are hard on average. However, in [13] one requires the function to be hard for circuits that have a SATISFIABILITY oracle whereas Theorem 5 does not. Even though we require lower bounds against a weaker class of circuits, our result is incomparable to that of [13] as the quantities that appear in the two theorems are different and in particular the simulation of [13] succeeds on a significantly larger fraction of the inputs.

C. Organization of the paper

Due to space limitations this extended abstract is missing some details and does not contain precise proofs. The precise details and full proofs can be found in the full version of this paper. We give a high level overview of the technique of this paper in Section II. This overview sketches how to prove our main results. In Sections III,IV we introduce some notation and state our main general theorem that shows how to use seedless extractors to get derandomization. We finish with some open problems in Section V.

II. Overview of the technique

In this section we give a high level informal overview of the main ideas that are used in our proof. The reader is referred to the technical section for precise details.

A. Extracting randomness from the input

The goal of achieving what we call "weak derandomization" was considered by Goldreich and Wigderson [13]. They consider randomized algorithms in which the number of random coins is smaller than the input length. Their high level idea is that in such cases one can try and "extract" randomness r from the input x. An obvious difficulty is that this makes the input and random coins correlated. Consider for example a randomized algorithm $A(x, r)$ where $|x| = |r|$ and define the deterministic algorithm $B(x) = A(x, x)$. It may be the case that for every x, all $r \neq x$ have $A(x, r) = f(x)$ while $A(x, x) \neq f(x)$ and then B errs on every input. Goldreich and Wigderson restrict their attention to randomized algorithms in which there exist many coin outcomes r that are each good for all inputs x. Surprisingly, they show that there are interesting algorithms with this property in the literature. We remark that the construction presented in [13] also uses "seeded extractors" to amplify

the success probability of A.[8]

Zimand's result [50] (see Theorem 2) also uses the approach of extracting randomness from the input. This is done using a new variant of seeded extractors (called "exposure resilient extractors" [49]). Zimand shows that using the approach of Goldreich and Wigderson with exposure resilient extractors gives an explicit version of Yao's lemma for decision trees.

B. Our approach

In this paper we develop a general technique to prove explicit versions of Yao's Lemma. Given a randomized algorithm A with $m \leq n$ coins and a function $E : \{0,1\}^n \rightarrow \{0,1\}^m$ we identify properties of E that are sufficient to conclude that the deterministic algorithm $B(x) = A(x, E(x))$ "weakly derandomizes" A. A little bit more precisely, we show that any class of randomized algorithms defines a class of probability distributions such that if E is a (seedless) extractor with very small error for the class of distributions (meaning that for any distribution X in the class, $E(X)$ has statistical distance less than 2^{-m} from the uniform distribution on m bit strings) then the deterministic algorithm $B(x) = A(x, E(x))$ weakly derandomizes any randomized algorithm A in the class.

We stress that our analysis is different than the analysis of [13] or [50]. In particular, the distributions that come up in the proofs of [13], [50] *depend* on the extractor (and therefore do not allow seedless extraction). On the other hand, we need extractors with very small error that cannot be achieved by seeded extractors with short seeds. We therefore must use seedless extractors.

We state and prove the general theorem in Section IV and our results follow by using appropriate explicit extractors. For communication protocols and decision trees we use "off the shelf" seedless extractor constructions (we use "2-source extractors' for the former and "bit-fixing source extractors" for the latter). For constant depth algorithms and general algorithms we introduce and construct a new variant of seedless extractors which we call "extractors for recognizable distributions". We elaborate on these extractors below.

C. The argument for communication protocols

We now give a high level overview of our analysis. For concreteness we focus on proving Theorem 1 on communication protocols. We later explain how to generalize the argument to other setups.

The setup: We are given a randomized (public coin) communication protocol $A(x, r)$ with communication complexity q and m random coins, such that A computes some function $f : \{0,1\}^n \rightarrow \{0,1\}$ with success $1 - \rho \geq 2/3$. In this setting we think os an input $x \in \{0,1\}^n$ as a pair $x = (x_1, x_2)$ where $x_1, x_2 \in \{0,1\}^{n/2}$. We use m to denote the number of coins used by A. That is, $m = |r|$. (The reader may think of $q, m = \text{polylog}(n)$ in order to avoid having many parameters). To weakly derandomize A we use a 2-source extractor $E(x) = E(x_1, x_2)$. We now explain what are 2-source extractors.

Given a set $S \subseteq \{0,1\}^n$ we use U_S to denote the uniform distribution over S. A set $S \subseteq \{0,1\}^{n/2} \times \{0,1\}^{n/2}$ is a *rectangle* if $S = S_1 \times S_2$ for some $S_1, S_2 \in \{0,1\}^{n/2}$. A 2-source extractor is a function $E : \{0,1\}^{n/2} \times \{0,1\}^{n/2} \rightarrow \{0,1\}^m$ such that for every "sufficiently large" rectangle $S \subseteq \{0,1\}^{n/2} \times \{0,1\}^{n/2}$ the distribution $E(U_S)$ is statistically close to the uniform distribution.[9] We use "off the shelf" explicit constructions of 2-source extractors [7], [45], [9]. We remark that our proof critically uses the fact that the statistical distance can be smaller than 2^{-m} (which cannot be achieved using "seeded extractors" with seed length smaller than m [33], [34]). We consider the deterministic algorithm

$$B(x) = A(x, E(x))$$

Our goal is to show that:

- B succeeds on a $1 - \rho$ fraction of the inputs.
- B can be implemented by a communication protocol with complexity $O(q + m)$.

We start by showing the first item above and will elaborate on the second item later on. We remark that for the first item above it suffices to show that B succeeds on a say $1 - 4\rho$ fraction of the inputs as we can amplify the success probability of A from $1 - \rho$ to $1 - \rho/4$ before we start. (Note that this amplification only changes q and m by a constant factor).[10] Let us consider a probability space that consists of two independent random variables: X which is uniformly distributed over $\{0,1\}^n$ and R which is uniformly distributed over $\{0,1\}^m$. We define:

$$\begin{aligned} \alpha &= \Pr[A(X,R) = f(X)] \\ \beta &= \Pr[A(X, E(X) = f(X))] \end{aligned}$$

[8] More specifically, the deterministic algorithm presented in [13] is $B(x) = \mathsf{majority}_{y \in \{0,1\}^d} A(x, E(x,y))$ where $E : \{0,1\}^n \times \{0,1\}^{d=O(\log n)} \rightarrow \{0,1\}^m$ is an explicit (that is polynomial time computable) "seeded extractor" see e.g. [31], [38], [44] for survey articles on seeded extractors). The proof can be viewed as applying the aforementioned idea of $B(x) = A'(x,x)$ on the randomized algorithm $A'(x,r) = \mathsf{majority}_{y \in \{0,1\}^d} A(x, E(r,y))$ (which by [51] computes the same function computed by A with improved success probability).

[9] A more common definition of 2-source extractors considers distributions $X = (X_1, X_2)$ where X_1, X_2 are independent distributions with "sufficiently large min-entropy". It is standard that the two definitions are equivalent.

[10] This is the only place in the proof where we use the assumption that for every x, $\Pr_{R \leftarrow \{0,1\}^m}[A(x,R) = f(x)] \geq 1 - \rho$. The remainder of the argument only uses that $\Pr_{X \leftarrow \{0,1\}^n, R \leftarrow \{0,1\}^m}[A(X,R) = f(X)] \geq 1 - \rho$. In particular under the latter assumption we get a deterministic algorithm which succeeds with probability $1 - 3\rho - 2^{-m/10}$.

We note that by a straightforward averaging argument:

$$\begin{aligned}\alpha &= \Pr[A(X,R)=f(X)] \\ &= \sum_{x\in\{0,1\}^n} \Pr[A(x,R)=f(x)]\cdot\Pr[X=x] \geq 1-\rho\end{aligned}$$

Our goal is to show that β is not much smaller than α. The main difficulty is that in the definition of α, A is run using random coins R that are independent of X whereas in the definition of β, $E(X)$ completely depends on X. We should somehow deal with this correlation.

Intuition why 2-source extractors seem helpful: We have that for every $r \in \{0,1\}^m$ the function $A_r(x) = A(x,r)$ is a deterministic communication protocol of complexity q and therefore we can associate with it a function Q_r which maps inputs $x = (x_1, x_2)$ into the transcript of the protocol A_r on x. Let $Z = \{0,1\}^q$ be the set of possible transcripts. Note that for a fixed transcript $v \in Z$ and a fixed $r \in \{0,1\}^m$ the set $S_{r,v} = \{x : Q_r(x) = v\}$ is a rectangle and that the output of $A(\cdot, r)$ is fixed on this rectangle. As E is a 2-source extractor and the rectangle $S_{r,v}$ is large for "typical" choices of r, v we have that $E(U_{S_{r,v}})$ is statistically close to uniform. Loosely speaking, this means that the answer of $A(\cdot, r)$ is independent of $E(U_{S_{r,v}})$ and therefore we can hope to use the latter randomness in the protocol.

While the intuition above explains the relevance of 2-source extractors, we stress that the argument above seems circular and does not make much sense. This is because the set $S_{r,v}$ is defined as *a function of* the random coins r. In the next paragraph we explain how to make use of the intuition above and overcome the difficulty.

Analyzing the success probability of B: In order to make the intuition above work we examine the quantities α and β more closely. We express α as follows:

$$\begin{aligned}\alpha &= \Pr[A(X,R)=f(X)] \\ &= \sum_{r\in\{0,1\}^m} \Pr[A(X,r)=f(X)\wedge R=r] \\ &= \sum_{r\in\{0,1\}^m}\sum_{v\in Z} \Pr[A(X,r)=f(X)\wedge R=r\wedge Q_r(x)=v] \\ &= \sum_{r\in\{0,1\}^m}\sum_{v\in Z} \alpha_1(r,v)\cdot\alpha_2(r,v)\cdot\alpha_3(r,v)\end{aligned}$$

where

$$\begin{aligned}\alpha_1(r,v) &= \Pr[Q_r(X)=v] \\ \alpha_2(r,v) &= \Pr[R=r|Q_r(X)=v] \\ \alpha_3(r,v) &= Pr[A(X,r)=f(X)|Q_r(X)=v\wedge R=r]\end{aligned}$$

We can apply the same analysis on β (replacing R by $E(X)$) and conclude that:

$$\beta = \sum_{r\in\{0,1\}^m}\sum_{v\in Z} \beta_1(r,v)\cdot\beta_2(r,v)\cdot\beta_3(r,v)$$

where

$$\begin{aligned}\beta_1(r,v) &= \Pr[Q_r(X)=v] \\ \beta_2(r,v) &= \Pr[E(X)=r|Q_r(X)=v] \\ \beta_3(r,v) &= Pr[A(X,r)=f(X)|Q_r(X)=v\wedge E(X)=r]\end{aligned}$$

We need to show that $\alpha \approx \beta$ and for that we want to show that for "typical" r, v,

$$\alpha_1(r,v)\cdot\alpha_2(r,v)\cdot\alpha_3(r,v) \approx \beta_1(r,v)\cdot\beta_2(r,v)\cdot\beta_3(r,v)$$

Note that for every r, v we have that $\alpha_1(r,v) = \beta_1(r,v)$. Furthermore as R is independent of X we can simplify α_2, α_3 and get that for every r, v,

$$\alpha_2(r,v) = \Pr[R=r|Q_r(X)=v] = \Pr[R=r] = 2^{-m}$$

$$\begin{aligned}\alpha_3(r,v) &= \Pr[A(X,r)=f(X)|Q_r(X)=v\wedge R=r] \\ &= \Pr[A(X,r)=f(X)|Q_r(X)=v]\end{aligned}$$

The difficulty is that $E(X)$ *does depend* on X and we cannot repeat these steps for β. Still, we will be able to relate α_2 to β_2 using the fact that E is a 2-source extractor. More precisely, for a pair r, v the set $S_{r,v} = \{x : Q_r(x) = v\}$ is a rectangle and by a Markov argument for typical r, v the rectangle is "large" (that is of size roughly 2^{n-q}). When E is applied on $U_{S_{r,v}}$ it produces a distribution that is ϵ-close to uniform. We use 2-source extractors with very small error (say $\epsilon = 2^{-100m}$) and with this setting we get:

$$\begin{aligned}\beta_2(r,v) &= \Pr[E(X)=r|Q_r(X)=v] = \Pr[E(U_{S_{r,v}})=r] \\ &= 2^{-m} \pm \epsilon \approx 2^{-m} = \alpha_2(r,v)\end{aligned}$$

Thus, we will be done if we show that for "typical" r, v it holds that $\alpha_3(r,v) \approx \beta_3(r,v)$. Unfortunately, this does not seem to follow. Note that for fixed r, v, the quantity $\alpha_3(r,v)$ measures the success probability of $A_r(\cdot)$ on a uniformly chosen input in $S_{r,v}$. On the other hand $\beta_3(r,v)$ measures the success probability of $A_r(\cdot)$ on a uniformly chosen input in $S'_{r,v} = S_{r,v} \cap \{x : E(x) = r\}$. We know that $A_r(\cdot)$ is constant over $S_{r,v}$ (as the output of the communication protocol is a function of the transcript v). However, note that $S'_{r,v}$ is much smaller than $S_{r,v}$. We just saw that $|S'_{r,v}|/|S_{r,v}| \approx 2^{-m}$. It may be the case that the set $\{x \in S_{r,v} : A_r(x) \neq f(x)\}$ is very small and still contains $S'_{r,v}$. When this occurs, $\alpha_3(r,v)$ may be significantly larger than $\beta_3(r,v)$.

In order to overcome this difficulty we modify the argument above as follows. We observe that by the (non-explicit) version of Yao's lemma there exists $r' \in \{0,1\}^m$ such that setting $f'(x) = A(x,r')$ we have that $\Pr[f'(X) = f(X)] \geq 1-\rho$. We consider modified definitions of α and β where the function f is replaced with the function f': That is

$$\begin{aligned}\alpha' &= \Pr[A(X,R)=f'(X)] \\ \beta' &= \Pr[A(X,E(X)=f'(X))]\end{aligned}$$

Note that $|\alpha' - \alpha| \leq \rho$ and $|\beta' - \beta| \leq \rho$. Thus, to achieve our goal and show that β is not much smaller than α it suffices to show that β' is not much smaller than α'.

All the calculations made so far are oblivious to whether we use f or f' in the definitions of α and β. We are going to repeat all the calculations above using α' and β'. We also introduce the following modification: Instead of using the function Q_r to partition $\{0,1\}^n$ into 2^q rectangles, we use the function $Q'_r(x) = (Q_r(x), Q_{r'}(x))$ to partition $\{0,1\}^n$ into 2^{2q} rectangles. Once again, the calculations above are oblivious to the difference between Q_r and Q'_r and follow in precisely the same way with the modifications we have added. In particular, just like before it follows that $\alpha'_2(r,v) \approx \beta'_2(r,v)$ for a typical pair (r,v). The advantage of the modification is that when we consider α'_3 and β'_3 we now have that:

$$\begin{aligned}\alpha'_3(r,v) &= \Pr[A(X,r) = f'(X) | Q'_r(X) = v \wedge R = r] \\ &= \Pr[A(X,r) = f'(X) | Q'_r(X) = v]\end{aligned}$$

$$\beta'_3(r,v) = \Pr[A(X,r) = f'(X) | Q'_r(X) = v \wedge E(X) = r]$$

However, conditioned on the event $\{Q'_r(X) = v\}$ we have that both $Q_r(X)$ and $Q_{r'}(X)$ are fixed. Recall that $Q_r(X)$ determines $A(X,r)$ and $Q_{r'}(X)$ determines $f'(X) = A(X,r')$. Therefore we get that the two random variables $A(X,r)$ and $f'(X)$ are fixed when conditioned on the event $\{Q'_r(X) = v\}$. Let us compare α'_3 and β'_3. The difference is that definition of β'_3 contains an "additional conditioning" to the event $\{E(X) = r\}$. However, the random variables that appear in the probability were fixed even before this additional conditioning. Thus, the additional conditioning cannot change their values and for every r, v, $\alpha'_3(r,v) = \beta'_3(r,v)$. To conclude, we showed that for typical r, v have that

$$\alpha'_1(r,v) \cdot \alpha'_2(r,v) \cdot \alpha'_3(r,v) \approx \beta'_1(r,v) \cdot \beta'_2(r,v) \cdot \beta'_3(r,v)$$

It follows that $\alpha' \approx \beta'$ and therefore $\alpha \approx \beta$ as required.

The communication complexity of B: We need to show that we can implement the function $B(x) = A(x, E(x))$ using a deterministic communication protocol of complexity $O(q+m)$. It suffices to construct a 2-source extractor $E(x_1, x_2)$ that can be implemented by a deterministic communication protocol with complexity $O(q+m)$, as the two parties can first compute $E(x_1, x_2)$ and then simulate $A(x_1, x_2)$ using this value as random coins.

The key observation is that we are interested in very large rectangles, namely rectangles of size $2^{n-\Delta}$ for $\Delta = O(q+m)$. Alternatively, when using the "min-entropy formulation" of 2-source extractors this corresponds to two independent sources where the sum of min-entropies of the two sources is at least $n - \Delta$.

In this regime of parameters it is possible top have an extractor E that depends only on $O(\Delta)$ bits of its n bit input. More specifically, following [12], [37] we use the construction:

$$E(x_1, x_2) = E'(x_1|_{1,\ldots,3\Delta}, x_2|_{1,\ldots,3\Delta})$$

where E' is an "off the shelf" explicitly constructible 2-source extractor for rectangles of size $2^{2n/3}$ [7], [45], [9]. It is not hard to see that E is indeed a 2-source extractor for rectangles of size $2^{n-\Delta}$. Intuitively, each of the two sources is missing at most Δ "bits of entropy" and therefore a 3Δ bit slice has entropy rate at least $2/3$. Obviously, E has a deterministic communication protocol with complexity $6\Delta = O(q+m)$ in which each party sends a prefix of length 3Δ of his input. Finally, B is explicitly constructible as both A and E are explicitly constructible.

D. Extending the argument to other classes of algorithms

Decision trees: The argument of the previous section also applies to decision trees with few modifications. In this setup $Q_r(x)$ will be a leaf that the decision tree $A(\cdot, r)$ gets to on input x. We now need extractors that can extract from distributions which are uniform over sets of the form $\{x : Q_r(x) = v\}$. In such a distribution the bits that were queried on the path to the leaf are fixed while the other bits are uniformly distributed and independent. In the literature on extractors such distributions are referred to as "bit-fixing sources". There are explicit constructions of seedless extractors for bit-fixing sources (see e.g. [8], [22], [11], [35]) and using our analysis gives an explicit version of Yao's lemma for decision trees using these extractors.

Streaming algorithms: We can also apply the approach to streaming algorithms. In this setup $Q_r(x)$ is the final state of the streaming algorithm $A(\cdot, r)$ on input x. We use specially designed extractors (that we construct using 2-source extractors) to extract randomness from distributions which are uniform over sets of the form $\{x : Q_r(x) = v\}$. The high level idea resembles the techniques used in extractors for sources samplable by small width branching programs (see e.g. [25], [21]).

1) Extractors for recognizable distributions: We can generalize the approach further to any class of randomized algorithms. For this purpose we introduce a new notion of seedless extractors that we call "extractors for recognizable distributions". This new notion of extractors resembles "extractors for samplable distributions" defined by Trevisan and Vadhan [41]. This new notion is very natural and to the best of our knowledge was not explicitly defined before.

Let $\mathcal{C}$ be a class of functions $C : \{0,1\}^n \to \{0,1\}$. We say that $E : \{0,1\}^n \to \{0,1\}^m$ is a (k,ϵ)-extractor for distributions recognizable by $\mathcal{C}$ if for every $C \in \mathcal{C}$ such that $S_C = \{x : C(x) = 1\}$ is of size at least 2^k, $E(U_{S_C})$ is ϵ-close to the uniform distribution. (Recall that U_{S_C} is the uniform distribution on S_C).

We remark that several families of distributions that were considered in the literature on extractors are naturally

captured by this notion: For example, bit fixing sources are distributions recognizable by projections and affine sources are distributions recognizable by affine functions.

The general case: The analysis we described in Section II-C can be generalized as follows: Let $\mathcal{A}$ be a class of randomized algorithms $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ (with some natural closure properties) and let

$$\mathcal{C} = \{A(\cdot, r) : A \in \mathcal{A}, r \in \{0,1\}^m\}$$

namely the set of all functions $C(x)$ such that there exist A and r such that $C(x) = A(\cdot, r)$. Let E be an extractor for distributions recognizable by $\mathcal{C}$ for appropriate parameters k, ϵ. Then, for every randomized algorithm $A \in \mathcal{A}$ the deterministic algorithm $B(x) = A(x, E(x))$ weakly derandomizes A. Indeed, 2-source extractors can be seen as extractors for distributions recognizable by communication protocols, and extractors for bit-fixing sources can be seen as extractors for distributions recognizable by decision trees.

Uniform AC^0 algorithms: Our explicit version of Yao's lemma for AC^0 follows by constructing extractors for distributions recognizable by AC^0 circuits. We observe that the parity function on n bits (which we denote by $p_n(x)$) is an extractor for distributions recognizable by AC^0 circuits. This follows using Hastad's result [14] that parity is hard on average for AC^0. We need extractors that extract more than one bit and show that applying the parity function m times on disjoint blocks of the input gives extractors that output m bits. (We remark that the extractors we get only extract from sources with min-entropy $k = n - n^{\Omega(1)}$, but this is sufficient for our application).

Theorem 4 follows by using our general analysis with the extractor mentioned above. In this setup we can handle randomized algorithms with a polynomial number of random coins. This is done by first using pseudorandom generators for AC^0 [32], [24] to reduce the number of random coins used so that it is polylogarithmic in the input length. Once the number of random coins is small we can use our extractors and get weak derandomization of uniform randomized AC^0 algorithms.

It is natural to compare this approach with the approach of [32] that give strong derandomization in quasi-polynomial time. This result is obtained by using the same first step, namely reducing the number of random bits using pseudorandom generators (and in fact, the construction of pseudorandom generators is the main technical contribution of that paper). In the second step the algorithm is derandomized by running it on all possible choices for random coins (that is on all seeds of the generator) and computing the majority vote. This takes time that is exponential in the number of seeds and gives a quasi-polynomial time algorithm. In contrast, after reducing the number of coins, we run the algorithm only once and this is why we obtain a polynomial time algorithm.

General polynomial time algorithms: Our results on AC^0 algorithms use the fact that parity is hard for AC^0. By abstracting the properties of parity that are used in the proof we get the conditional results mentioned in Theorem 5.

III. Preliminaries

General notation: We use $x \circ y$ to denote the concatenation of two strings x, y. We use $[n]$ to denote the set $\{1, \ldots, n\}$. Given a string $x \in \{0,1\}^n$ and a set $S \subseteq [n]$ we use x_S to denote the restriction of x to S (which is a string of length $|S|$).

Probability distributions: Given a probability distribution P we use $X \leftarrow P$ to denote the experiment in which X is chosen at random according to P. Given a set S we use $X \leftarrow S$ to denote the experiment in which X is chosen uniformly at random from S. We use U_m to denote the uniform distribution on $\{0,1\}^m$.

We say that two distributions P, Q on a set Ω are ϵ-close if $\frac{1}{2} \cdot \sum_{x \in \Omega} |\Pr[P = x] - \Pr[Q = x]| \leq \epsilon$. The min-entropy of a distribution X on Ω is defined by $H_\infty(X) = min_{x \in \Omega} \frac{1}{\Pr[X = x]}$.

IV. Main technical theorem

In this section we state and prove our main technical theorem. Our goal is to take a randomized algorithm $A(x, r)$ and show that $B(x) = A(x, E(x))$ is a deterministic algorithm that does as well on a random input. Theorem 10 below states conditions on E that are sufficient. We need the following definitions.

Definition 6: Let $\mathcal{Q}$ be a family of functions $Q : \{0,1\}^n \to \{0,1\}^q$. We say that a function $B : \{0,1\}^n \to \{0,1\}$ is *determined by* $\mathcal{Q}$ if there exist functions $Q \in \mathcal{Q}$ and $D : \{0,1\}^q \to \{0,1\}$ such that for every $x \in \{0,1\}^n$, $B(x) = D(Q(x))$.

Given a function $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ and $r \in \{0,1\}^m$ we define $A_r : \{0,1\}^n \to \{0,1\}$ by $A_r(x) = A(x, r)$. We say that a function $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ is *determined by* $\mathcal{Q}$ if for every $r \in \{0,1\}^m$ the function A_r is determined by $\mathcal{Q}$.

In order to make the definition above less abstract note for example that any deterministic communication protocol $B(x)$ of communication complexity q is determined by $Q(x)$ which is the q bit transcript of the protocol on x. Furthermore, let $\mathcal{Q}$ be the set of all functions $Q : \{0,1\}^n \to \{0,1\}^q$ that describe the transcripts of deterministic communication protocols of communication complexity q, then every randomized communication protocol $A(x, r)$ of communication complexity q is determined by $\mathcal{Q}$. Similarly, every deterministic decision tree $B(x)$ is determined by $Q(x)$ which is the leaf that is obtained on x.

Note that we can choose different families $\mathcal{Q}$ for the same function B. For example, any communication protocol (in fact any function) $B(x)$ is determined by itself, that is by the function $Q(x) = B(x)$.

Given a function $Q : \{0,1\}^n \to \{0,1\}^q$ and $v \in \{0,1\}^q$ we define a probability distribution as follows:

Definition 7: Let $Q : \{0,1\}^n \to \{0,1\}^q$ be a function and let $v \in \{0,1\}^q$. We define $S_{Q=v} = \{x \in \{0,1\}^n : Q(x) = v\}$ and $P_{Q=v}$ to be the uniform distribution over $S_{Q=v}$.

The next definition discusses functions $\bar{Q}$ that are obtained by concatenating two copies of functions $Q \in \mathcal{Q}$.

Definition 8 (concatenation of functions): Let $\mathcal{Q}$ be a family of functions $Q : \{0,1\}^n \to \{0,1\}^q$ and let t be an integer. We say that a function $\bar{Q} : \{0,1\}^n \to \{0,1\}^{2q}$ is a *concatenation* of functions in $\mathcal{Q}$ if there exist $Q_1, Q_2 \in \mathcal{Q}$ (not necessarily distinct) such that for every $x \in \{0,1\}^n$

$$\bar{Q}(x) = Q_1(x) \circ Q_2(x)$$

Let $\mathcal{Q}^2$ denote the class of all concatenations of functions in $\mathcal{Q}$.

We will be interested in "seedless extractors" that extract randomness from any distribution that is associated with a function $Q \in \mathcal{Q}$ and $v \in \{0,1\}^q$. In order to apply our argument we will need to consider a slightly stronger variant (defined below) which extracts randomness also from distributions that are obtained by concatenating functions from $\mathcal{Q}$.

Definition 9 (extractors against $\mathcal{Q}$): Let $\mathcal{Q}$ be a family of functions $Q : \{0,1\}^n \to \{0,1\}^q$ and let t, k be integers. We say that a function $E : \{0,1\}^n \to \{0,1\}^m$ is a (k,ϵ)-*extractor against concatenations of* $\mathcal{Q}$ if for every function $\bar{Q} \in \mathcal{Q}^2$ and $v \in \{0,1\}^{2q}$ such that $H_\infty(P_{\bar{Q}=v}) \geq k$ we have that $E(P_{\bar{Q}=v})$ is ϵ-close to U_m.

We are finally ready to state our main technical theorem. Loosely speaking, the theorem says that if a randomized algorithm A is determined by a family $\mathcal{Q}$ then we can use an extractor against concatenations of $\mathcal{Q}$ to construct a deterministic algorithm $B(x) = A(x, E(x))$ that "weakly derandomizes" A.

Theorem 10: There exists a constant $c > 0$ such that for every m, n the following holds: Let $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a function that is determined by a family $\mathcal{Q}$ of functions $Q : \{0,1\}^n \to \{0,1\}^q$. Let $k = n - 2q - 100m$ and $\epsilon = 2^{-100m}$. Let $E : \{0,1\}^n \to \{0,1\}^m$ be an (k,ϵ)-extractor against concatenations of $\mathcal{Q}$. Let $\rho \leq 1/3$ and let $f : \{0,1\}^n \to \{0,1\}$ be some function and assume that

$$\Pr_{X \leftarrow U_n, R \leftarrow U_m}[A(X,R) = f(X)] \geq 1 - \rho$$

Then

$$\Pr_{X \leftarrow U_n}[A(X, E(X)) = f(X)] \geq 1 - 3\rho - 2^{-10m}$$

The statement of Theorem 10 is somewhat technical. We now outline an alternative interpretation of the theorem that we find more intuitive. This (weaker) formulation resembles the high level overview presented in Section II-D1. We stress that the remainder of the paper relies only on the formulation in the theorem and the interpretation below is included for explanatory purposes.

Remark 11: (*Interpretation using "recognizable distributions"*): Let $\mathcal{C}$ be a "complexity class" namely a set of functions $C : \{0,1\}^n \to \{0,1\}$. We assume that $\mathcal{C}$ is closed under parity, namely that for every $C_1, C_2 \in \mathcal{C}$ the function $C(x) = C_1(x) \oplus C_2(x)$ is also in $\mathcal{C}$. Let $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a randomized algorithm and let $f : \{0,1\}^n \to \{0,1\}$ be a function such that

$$\Pr_{X \leftarrow U_n, R \leftarrow U_m}[A(X,R) = f(X)] \geq 1 - \rho$$

We assume that A is computable by the class $\mathcal{C}$ in the sense that for every $r \in \{0,1\}^m$ the function $C(x) = A(x,r)$ is in $\mathcal{C}$. Let $E : \{0,1\}^n \to \{0,1\}^m$ be an "extractor for distributions recognizable by $\mathcal{C}$". By that we mean that for every $C \in \mathcal{C}$ such that $S_C = \{x : C(X) = 1\}$ is of size at least 2^{n-102m} the experiment of sampling uniformly $x \leftarrow S_C$ and computing $E(x)$ produces a probability distribution that is 2^{-100m}-close to uniform. Then

$$\Pr_{X \leftarrow U_n}[A(X, E(X)) = f(X)] \geq 1 - 3\rho - 2^{-10m}$$

The latter formulation follows from Theorem 10 setting $q = 1$ and $\mathcal{C} = \mathcal{Q}$.

We remark that Theorem 10 can be extended in several ways. We discuss possible extensions in the full version.

V. Conclusion and open problems

In this paper we present a general technique to prove explicit versions of Yao's lemma and apply it in several computational settings. A natural research direction is to try and apply this technique to other classes of randomized algorithms.

The notion of weak derandomization that we consider only requires that the deterministic algorithm succeeds with high probability when the input is chosen according to the uniform distribution. A stronger notion considered in [18], [42] requires the deterministic algorithm to succeed with high probability on every distribution that is efficiently samplable. We remark that the approach of this paper can be extended to allow every high min-entropy distribution that is "recognizable" in the sense explained in Remark 11. It is interesting to try and extend our approach to other families of distributions.

We remark that subsequent to this work Salil Vadhan and Dieter van Melkebeek suggested an alternative proof for Theorems 4,5. This approach is described and further developed by Jeff Kinne, Dieter van Melkebeek and the author in [23]. One of the consequences is an improved version of Theorem 5 that gives weak derandomization of every language in BPP starting from a weaker assumption than the one given in Theorem 5. Specifically, it is sufficient that the function h_n in Theorem 5 is only "mildly hard on average" in the sense that every circuit C of size n^c

attempting to compute h_n errs on a $1/n$ fraction of the inputs.

ACKNOWLEDGMENTS

This research was supported by BSF grant 2004329 and ISF grant 686/07.

I am grateful to Oded Goldreich, Chris Umans, Salil Vadhan, Dieter van Melkebeek and Avi Wigderson for interesting discussions on "weak derandomization". I am grateful to Eyal Kushilevitz and Ilan Newman for interesting discussions on communication complexity and decision trees.

REFERENCES

[1] L. Adelman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, 1978.

[2] M. Ajtai. Approximate counting with uniform constant depth circuits. *Advances in computational complexity theory*, pages 1–20, 1990.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[4] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[5] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, Nov. 1984.

[6] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[7] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, Apr. 1988. Special issue on cryptography.

[8] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 396–407, 1985.

[9] Y. Dodis, A. Elbaz, R. Oliveira, and R. Raz. Improved randomness extraction from two independent sources. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 334–344, 2004.

[10] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. Testing and spot-checking of data streams (extended abstract). In *SODA*, pages 165–174, 2000.

[11] A. Gabizon, R. Raz, and R. Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. *SICOMP: SIAM Journal on Computing*, 36(4):1072–1094, 2006.

[12] O. Goldreich and A. Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997.

[13] O. Goldreich and A. Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA*, pages 209–223, 2002.

[14] J. Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20. ACM, 1986.

[15] R. Impagliazzo. Can every randomized algorithm be derandomized? In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 373–374, 2006.

[16] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, 1994.

[17] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[18] R. Impagliazzo and A. Wigderson. Randomness vs. time: Derandomization under a uniform assumption. In *39th Annual Symposium on Foundations of Computer Science*. IEEE, 1998.

[19] V. Kabanets. Derandomization: A brief overview. In *Electronic Colloquium on Computational Complexity, technical reports, TR 02-008*, 2002.

[20] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[21] J. Kamp, A. Rao, S. Vadhan, and D. Zuckerman. Deterministic extractors for small-space sources. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 691–700, 2006.

[22] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput*, 36(5):1231–1247, 2007.

[23] J. Kinne, D. van Melkebeek, and R. Shaltiel. Pseudorandom generators and typically-correct derandomizatiom. *Submitted*, 2009.

[24] A. Klivans. On the derandomization of constant depth circuits. In M. X. Goemans, K. Jansen, J. D. P. Rolim, and L. Trevisan, editors, *RANDOM-APPROX*, volume 2129 of *Lecture Notes in Computer Science*, pages 249–260. Springer, 2001.

[25] R. König and U. M. Maurer. Generalized strong extractors and deterministic privacy amplification. In *IMA Int. Conf.*, pages 322–339, 2005.

[26] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[27] P. B. Miltersen. Derandomizing complexity classes. In *Handbook of Randomized Computing, Kluwer*, pages 843–941, 2001.

[28] I. Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.

[29] N. Nisan. Crew prams and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991.

[30] N. Nisan. Pseudorandom generators for space bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[31] N. Nisan and A. Ta-Shma. Extracting randomness: A survey and new constructions. *JCSS: Journal of Computer and System Sciences*, 58, 1999.

[32] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, Oct. 1994.

[33] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[34] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[35] A. Rao. Extractors for low weight affine sources. *Unpublished Manuscript*, 2008.

[36] R. Raz and O. Reingold. On recycling the randomness of states in space bounded computation. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 159–168, 1999.

[37] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[38] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[39] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001.

[40] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the xor lemma. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, 1999.

[41] L. Trevisan and S. Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[42] L. Trevisan and S. P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[43] C. Umans. Pseudo-random generators for all hardnesses. In *Proceedings of the Thirty-fourth Annual ACM Symposium on the Theory of Computing*, 2002.

[44] S. Vadhan. Randomness extractors and their many guises. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 9–12, 2002.

[45] U. Vazirani. Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources. *Combinatorica*, 7:375–392, 1987.

[46] A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 Nov. 1982. IEEE.

[47] A. C.-C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing, 30 April-2 May, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979.

[48] A. C.-C. Yao. Lower bounds by probabilistic arguments (extended abstract). In *24th Annual Symposium on Foundations of Computer Science, 7-9 November 1983, Tucson, Arizona, USA*, pages 420–428, 1983.

[49] M. Zimand. Exposure-resilient extractors. In *IEEE Conference on Computational Complexity*, pages 61–72, 2006.

[50] M. Zimand. On derandomizing probabilistic sublinear-time algorithms. In *IEEE Conference on Computational Complexity*, pages 1–9, 2007.

[51] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11:345–367, 1997.