Towards proving strong direct product theorems

Ronen Shaltiel
Hebrew University, Jerusalem, Israel.
and
Institute for advanced study, Princeton, NJ.
E-mail:ronens@cs.huji.ac.il.

Abstract

A fundamental question of complexity theory is the direct product question. A famous example is Yao's XOR-lemma, [19] in which one assumes that some function f is hard on average for small circuits, (meaning that every circuit of some fixed size s which attempts to compute f is wrong on a non-negligible fraction of the inputs) and concludes that every circuit of size s' has a small advantage over guessing randomly when computing $f^{\oplus k}(x_1, \cdots, x_k) = f(x_1) \oplus \cdots \oplus f(x_k)$ on independently chosen $x_1, \cdots, x_k$. All known proofs of this lemma, [12, 6, 8, 5] have the feature that $s' < s$. In words, the circuit which attempts to compute $f^{\oplus k}$ is **smaller** *than the circuit which attempts to compute f on a single input!*

This paper addresses the issue of proving **strong** *direct product assertions, that is ones in which $s' \approx ks$ and is in particular* **larger** *than s. Since we are unable to "handle" boolean circuits we follow a direction suggested by previous works [13, 15] and study this question in weaker computational models such as decision trees and communication complexity games.*

Our first result is a counterexample which shows that strong direct product assertions are simply not true. This holds for every "reasonable" model of computation. While this counterexample rules out the possibility of proving strong direct product assertions, it seems to exploit defects in the formulation of the problem rather than show that our general intuition for strong direct product assertions is false. Nevertheless, in order to prove theorems with a strong direct product flavor we must change the formulation of the question and either strengthen the assumption or weaken the conclusion.

An example of strengthening the assumption is given in our main result in which we prove that if a function f is proved to be hard on average for c-bit communication protocols via the "discrepancy method" then $f^{\oplus k}$ is exponentially harder on average for $\Omega(kc)$-bit communication protocols. This follows by showing that $disc(f^{\oplus k}) = O(disc(f))^{\Omega(k)}$ which we prove using algebraic techniques inspired by [14]. The main technical step in the proof may be of independent interest and establishes a polynomial relation between the discrepancy of a matrix and its spectral norm.

As for weakening the conclusion, we introduce the notion of **fair** *algorithms which restricts the algorithm attempting to compute $f^{\oplus k}$ to spend its computational resource evenly between the k instances. Our third result is an optimal direct product theorem for fair decision trees.*

It is our hope that the notions suggested in this paper can be extended to stronger computational models.

1. Introduction

1.1. The direct product question

Suppose you are given a biased coin, (that is one in which the probability to get "heads" is $1/2+\alpha$) and you toss it k times and compute the exclusive-or of the outcomes. It is easy to see that the bias of the "new coin" you obtain goes to zero exponentially fast with k. (More precisely, the "new coin" will have probability $1/2 + (2\alpha)^k/2$).[1] The direct product question asks to what extent is this true in the computational world.

Suppose you have a boolean function which is hard on average for some complexity class. This intuitively means that from the point of view of any algorithm from that class the outcome of the function on a uniformly chosen input looks like a (biased) random coin. Is it true that xoring k independent copies of this "biased coin" produces a bit which is much less biased from the point of view of algorithms in the given complexity class? Such assertions are referred to as direct product assertions. It turns out that proving such assertions is much more involved than one might expect at first glance.

1.1.1 Statement of the question

Given a boolean function f over domain X and an integer k, we define a function $f^{\oplus k} : X^k \to \{0,1\}$.

$$f^{\oplus k}(x_1, \cdots, x_k) = f(x_1) \oplus \cdots \oplus f(x_k)$$

The intuition presented before suggests that if f is hard on average, (say f can be computed correctly by some complexity class on at most a $(1/2 + \alpha)$-fraction of of the inputs), then we expect $f^{\oplus k}$ to be computed correctly on a fraction of inputs which is about $(1/2+\alpha^k)$. In other words, we expect $f^{\oplus k}$ to become "exponentially harder" on average. What makes this problem much harder than the information theoretic problem about the exclusive-or of independent random coins presented above is that the algorithm attempting to compute $f^{\oplus k}(x_1, \cdots, x_k)$ may correlate computations on different inputs which intuitively corresponds to making the coins correlated. We will use an "abstract model of computation" to model the complexity class, as the direct product question can be stated in many computational models.

[1] A nice measure of the bias of a random coin is the *advantage* of the coin which is defined to be the difference between the probabilities of "heads" and "tails", or 2α in the notation above. The nice thing about this measure is that the advantage of the "new coin" is exactly the advantage of the original coin raised to the k'th power. To see this one encodes "heads" as 1 and tails as -1. In this encoding the advantage of a coin Z is given by $E(Z)$, and exclusive or is multiplication. It now follows that the advantage of $\prod Z_i$ is given by $E(\prod Z_i) = \prod E(Z_i)$.

1.1.2 An abstract model of computation

Consider some computational resource (such as circuit size, decision tree depth, number of bits exchanged in a communication protocol...). Let Res_r denote the class of all functions computable using r "units" of the resource. In this paper we consider the following classes:

- $Size_s$ the family of functions computed by circuits of size s.
- $Comm_c$ the family of functions (on two inputs) computed by a communication protocol which exchanges c bits.
- $Depth_d$ the family of functions computed by decision trees of depth d.

Saying that a function f is "hard on average" for Res_r means that every algorithm from Res_r computes f correctly on a bounded fraction of the inputs[2]. We use the following notation:

$$Suc_r^{Res}(f) = max_{P \in Res_r} \Pr_{x \in_R X}[P(x) = f(x)]$$

Since f is boolean it can always be computed correctly on at least half the inputs. Thus, we will be interested in the advantage the algorithm can get over guessing randomly which is given by: $Suc_r^{Res}(f) - 1/2$. It turns out that it is nicer to work with this quantity when it is normalized. We define the *advantage* Res_r has on f in the following way:

$$Adv_r^{Res}(f) = 2(Suc_r^{Res}(f) - 1/2)$$

When normalized this way, the advantage also has the following useful interpretation, (see also footnote 1).

$$Adv_r^{Res}(f) = max_{P \in Res_r} \Pr_{x \in_R X}[P(x) = f(x)] - \Pr_{x \in_R X}[P(x) \neq f(x)]$$

The direct product question can now be presented as follows:

The direct product problem: Is it true that for all f and r, k:

$$Adv_r^{Res}(f) \leq p \Rightarrow Adv_{r'}^{Res}(f^{\oplus k}) \leq p'$$

Where r' and p' are parameters which may depend on r, p and k. In words, one supposes that f is hard on average to algorithms with r units of the resource and concludes that $f^{\oplus k}$ is hard on the average to algorithms having r' units of the resource. Naturally, the assertion is stronger when r' is large and p' is small. It seems reasonable to allow the algorithm attempting to compute $f^{\oplus k}$ to use $r' = kr$

[2] In this paper we restrict ourselves to average case hardness relative to the uniform distribution. Some of out results do not generalize to arbitrary probability distributions. See the open problems in section 6.

units of the resource. This will at least enable it to run k copies of the best algorithm for f in Res_r on the k independent inputs. This strategy indeed computes f with advantage $Adv_r^{Res}(f)^k$. Thus, we say that the assertion is *optimal* when $r' = kr$ and $p' = p^k$.

However, it turns out that such assertions are often proven for much smaller r'. As we will see, in some cases only results with $r' << r << kr$ are known. In this paper we are interested in proving direct product assertions for *large* r'. We will call such assertions *strong* if $r' = \Omega(kr)$ and $p' = p^{\Omega(k)}$.

The strong-direct product problem: Is it true that for all f and r, k:

$$Adv_r^{Res}(f) \leq p \Rightarrow Adv_{\Omega(kr)}^{Res}(f^{\oplus k}) \leq p^{\Omega(k)}$$

The choice of $r' = \Omega(kr)$ is not that important. Most of our results give tradeoffs between r' and p'. What is important, (and different than most of the previous work in this area) is that we are interested in $r' >> r$.

Remark 1 *A different variant of the direct product problem that is often considered is the* concatenation variant. *It involves replacing the function $f^{\oplus k}$ with $f^{(k)}(x_1, \cdots, x_k) = (f(x_1), \cdots, f(x_k))$. In words, rather than trying to compute the exclusive-or of the outputs of the function on independent inputs, the algorithm is asked to compute* all *the outputs correctly simultaneously. In this setup the wanted assertion is the following:*

$$Suc_r^{Res}(f) \leq p \Rightarrow Suc_{r'}^{Res}(f) \leq p'$$

Once again, it is desired to have $r' = kr$ and $p' = p^k$, and one can define optimal and strong such assertions in an analogous way. The two variants of the direct product question are related. In particular, an optimal direct product theorem in the xor variant implies an optimal direct product theorem in the concatenation variant.

1.2. Previous work

The most studied computational model for direct product results is boolean circuits. The so called "Yao's XOR-lemma", [19] can be stated this way in our terminology:

$$Adv_s^{Size}(f) \leq p \Rightarrow Adv_{s'}^{Size}(f^{\oplus k}) \leq p^k + \epsilon$$

where $s' = s\left(\frac{\epsilon}{n}\right)^{O(1)}$, and n is the number of inputs of f. Note that in this result s' is actually *smaller* than s. In other words the circuit which tries to compute f on many instances is smaller than the one which tries to compute f on one instance. This is unavoidable in the sense that all known proofs of this lemma, [12, 6, 5, 8] work by proving the contra-positive claim: $Adv_{s'}^{Size}(f^{\oplus k}) > p^k + \epsilon \Rightarrow Adv_s^{Size}(f) > p$, and use the circuit which computes $f^{\oplus k}$ too well as a sub-circuit in the circuit that computes f too well. (See [5] for a survey on Yao's XOR-lemma).

Another weakness of this result is that p' is always larger than $1/s$ which means that one does not benefit from taking $k > \log s$. An unpublished result which is commonly attributed to Steven Rudich shows that all "black-box"[3] proofs of the XOR-lemma suffer from this flaw. Thus, proving a result in which $s' > s$ or $p' < 1/s$ seems to be beyond our current ability, as we do not know how to handle boolean circuits other than using them as black boxes. It should be noted that despite these weaknesses Yao's XOR-lemma has many applications in complexity theory.

The direct product question was also studied in other computational models. (We mention only previous work which is relevant to this paper, and the interested reader may find more references in [7]). Nisan et al, [13] study the direct product question in decision trees. They consider a specific variant of decision trees which they call "decision forests". A k-decision forest of depth d consists of k decision trees of depth d. Each is allowed to query all k inputs, and the i'th tree is supposed to compute $f(x_i)$. The final output of the decision forest is the concatenation of outputs of individual trees. Let us denote the class of all functions computable by depth d k-decision forests by $Forest_{k,d}$. With this terminology their result could be stated this way:

$$Suc_d^{Depth}(f) \leq p \Rightarrow Suc_{k,d}^{Forest}(f^{(k)}) \leq p^k$$

(Here $f^{(k)}(x_1, \cdots, x_k) = (f(x_1), \cdots, f(x_k))$, see remark 1). This result is optimal in the sense that a decision forest of depth d can run k decision trees of depth d in parallel and compute $f^{(k)}$ with success $Suc_d^{Depth}(f)^k$.

Parnafes et al, [15] used the technique of Raz's parallel repetition theorem [16] to prove a product theorem for "forests of c-bit communication protocols". (A forest of c-bit communication protocol is a collection of k c-bit communication protocols each is over all k inputs, and the i'th protocol is supposed to compute the function on the i'th input). Their result is similar in flavor to that of [13] with the exception that $p' = p^{\Omega(k/c)}$. This dependence on c comes from the technique of Raz, but whereas a dependence on c is unavoidable in the parallel repetition theorem (as was shown by [3]), it is open whether the result of [15] is best possible for forests of communication protocols.

1.3. Our results

Our first result is a general counterexample which shows that strong direct product assertions, (or even ones with r'

[3]"Black box" refers to proofs like the ones mentioned above, that use a circuit which computes $f^{\oplus k}$ too well as a black box in a circuit that computes f too well.

sufficiently larger than r) are simply not true. This counterexample applies to many models of computation and in particular to boolean circuits, communication protocols and decision trees.

While this counterexample rules out the possibility of proving strong direct product assertions, it seems to exploit defects in the formulation of the problem rather than show that our general intuition for direct product assertions is false. Intuitively, the algorithm of the counterexample is able to compute $f^{\oplus k}$ correctly with high probability by using its resources in an unbalanced way allocating a lot of its resources to specific instances. This does not contradict our intuition why strong direct product assertions are true as this is not a counterexample to our belief that it is not beneficial for the algorithm to correlate computations on different inputs. We elaborate on this point in section 3.3. In any case, as the assertion is not true as is, in order to capture our intuition and prove results with a strong direct product flavor we have to either strengthen the assumption or weaken the conclusion.

1.3.1 Strengthening the assumption: demanding more information on the function

The function presented in the counterexample has a large subset of "easy inputs", and it is feasible to check whether a given input is "easy". It is natural to ask what kind of restrictions can we place on the function in order to make a strong direct product assertion hold? We give such a restriction for communication protocols. This is done by insisting that the function f has low discrepancy. (The discrepancy of $f(x,y)$, denoted by $disc(f)$ measures how unbalanced is f in large rectangles). It is standard that low discrepancy entails that the function is hard on average for communication protocols. More precisely:

$$Adv_c^{Comm}(f) \leq disc(f)2^c$$

However, having low discrepancy is stronger than being hard on average and it intuitively says that the function has no large recognizable subset of easy inputs. The main result of this paper is the following inequality:

$$Adv_{kc/3}^{Comm}(f^{\oplus k}) \leq O(disc(f)2^c)^{k/3}$$

This inequality has the following interpretation: If the fact that $Adv_c^{Comm}(f) \leq p$ follows from the fact that f has low discrepancy ($disc(f) \leq p2^{-c}$) then a strong direct product assertion holds for f. We would like to point out that the "discrepancy method" is the most common way to prove that f is hard on average for communication protocols.

We prove the above statement by proving a "product theorem" for discrepancy. More precisely we prove that:

$$disc(f^{\oplus k}) = O\left(disc(f)\right)^{k/3}$$

The main step in this proof is establishing a connection between the discrepancy of a matrix and its spectral norm.

1.3.2 Discrepancy and spectral norm

A communication complexity problem f can be encoded as a matrix A where $A_{xy} = (-1)^{f(x,y)}$. The main lemma of the paper gives the following connection between the discrepancy of A and its spectral norm $||A||_2$:

$$\Omega\left(\frac{||A||_2}{N}\right)^3 \leq disc(A) \leq \frac{||A||_2}{N}$$

The proof is inspired by the methods of [14]. This lemma immediately implies the "product theorem" for discrepancy as $||A^{\otimes k}||_2 = ||A||_2^k$. Thus, once one can switch between discrepancy and spectral norm, the multiplicativity of the spectral norm gives the "product theorem" for discrepancy.

1.3.3 Weakening the conclusion: imposing restrictions on the algorithm

Another way to prove a strong direct product assertion is to weaken the conclusion. We suggest to do this by proving the assertion only for algorithms with certain restrictions. In a way the "forest model" of [13] is such a restriction. However, the forest model is only suitable for the "concatenation variant" of the direct product question and makes no sense in the "xor variant". In this paper we suggest a different restriction. The algorithm presented in the counterexample has the property that it uses its resource in an "unfair" way spending more than r units on particular inputs. We suggest to impose a "fairness" restriction on the algorithm. Some evidence to the potential of this direction is that we can prove an optimal direct product assertion for "fair" decision trees.

A decision tree of depth kd over variables $x_1, \cdots, x_k$ is *fair* if on every path from the root to a leaf at most d bits from each variable are queried. Let us denote the class of fair decision trees of depth kd by $FairDepth_{kd}$. It is not hard to prove the following result:

$$Adv_d^{Depth}(f) \leq p \Rightarrow Adv_{kd}^{FairDepth}(f^{\oplus k}) \leq p^k$$

It is our hope that the two directions we present here can be extended to prove strong direct product assertions for stronger computational models.

1.4. Organization of the paper

In section 3 we present our counterexample. In section 4 we prove our main result which gives a strong direct product assertion for communication protocol assuming the function has low discrepancy. In section 5 we prove a strong direct product theorem for fair decision trees. In section 6 we present open problems and possible generalizations of this paper.

2. Preliminaries

We use $\oplus$ to denote the exclusive or. For two matrices A and B of size $N \times N$, we use $A \otimes B$ to denote the *tensor product* of the two matrices. More precisely $A \otimes B$ is an $N^2 \times N^2$ matrix. We think of this matrix as an $N \times N$ matrix with entries being matrices of size $N \times N$ and place a copy of the matrix $A_{ij} \cdot B$ in the i'th row and j'th column. The tensor product of A with itself k times is denoted by $A^{\otimes k}$.

We use $Size_s$ to denote the class of all functions (over arbitrary number of inputs) computable by boolean circuits of size s.

We use $Comm_c$ to denote the class of all functions of two arguments which can be computed by a c-bit communication protocol. The exact definition of a communication protocol can be found in any textbook on this subject, (i.e. [11]). The only property of such protocols used in this paper is that such a protocol induces a partition of the inputs into 2^c rectangles.

We use $Depth_d$ to denote the class of all functions computed by a decision tree of depth d. Whereas a decision tree is a binary tree in which every internal node is labeled with a specific bit of the input, and leafs are labeled with outputs. An input to the decision tree defines a path from root to leaf in the obvious way, and the output of the tree on this input is the leaf label.

3. A general counterexample

In this section we give a general counterexample to direct product assertions with $r' >> r$ which works for boolean circuits, decision trees and communication protocols. We show that given a function which is hard given r units of the resource and easy given slightly more units, we can construct a function which is hard given r units of the resource, and yet computing it on k independent inputs is easy. We present the example using our general notation in section 3.1 and then draw conclusions for specific models in section 3.2. In section 3.3 we discuss the implications of this counterexample.

3.1. The general setting

We will present a function which is hard on average given r units of the resource, yet $f^{\oplus k}$ can be computed correctly with probability $1-2^{-k}$ given r' units of the resource, for r' sufficiently larger than r.

The counterexample works assuming the existence of a function which is hard given r units of the resource, and easy given slightly more units. Formally, we assume the existence of a function $g : \{0,1\}^n \to \{0,1\}$ and $r < \bar{r}$ such that:

- $Suc_r^{Res}(g) \leq 3/4$
- $g \in Res_{\bar{r}}$.

Another ingredient is an easy function (over few inputs) which answers one on a prescribed fraction of its inputs. Formally, Given a number $q < 1$ we assume the existence of a function $h : \{0,1\}^l \to \{0,1\}$ and a small number r^* such that

- $h \in Res_{r^*}$
- $\Pr_{y \in_R \{0,1\}^l}[h(y) = 1] = q$

Our counterexample function is a combination of the easy and hard functions.

Definition 1 *We define a function $f_q : \{0,1\}^n \times \{0,1\}^l \to \{0,1\}$ in the following way.*

$$f(x,y) = \begin{cases} g(x) & h(y) = 1 \\ 0 & h(y) = 0 \end{cases}$$

An algorithm for computing $f^{\oplus k}$ can utilize its resource smartly by spending a lot of the resource on the (expectedly few) inputs in which f involves the hard function, and spend a very small amount on other inputs. This is made formal in the following lemma.

"Lemma" 1 [4] *The following inequalities hold:*

- $Suc_r^{Res}(f) \leq 1 - q/4$
- *If* $r' \geq 2qk\bar{r} + kr^*$ *then* $Suc_{r'}^{Res}(f^{\oplus k}) \geq 1 - 2^{-\Omega(k)}$

Proof: (of lemma 1) For the first item note that an algorithm which is correct on f with probability greater than $1 - q/4$ must be correct on g with probability greater than $3/4$. For the second item, note that when $(y_1, \cdots, y_k)$ are randomly chosen, we expect qk of them to have $h(y_i) = 1$. By Chernoff's inequality the probability that more than $2qk$ of them have $h(y_i) = 1$ is bounded by $2^{-\Omega(k)}$. We can check which of the y_i's have $h(y_i) = 1$ using kr^* units of the resource. Assuming the constant function zero can be computed using 0 units of the resource, we can compute the

[4] The lemma is put in quotes because the formal statement requires some natural properties of the "abstract" computational model. Stating these properties precisely is tedious. The reader can verify that the algorithm described in the proof can be carried out in any of the models studied in this paper: boolean circuits, communication games and decision trees, (as well as any computational model that comes to mind).

function f on (x_i, y_i)'s such that $h(y_i) = 0$. Assuming that there are at most $2qk$ of y_i's such that $h(y_i) = 1$ we can use $2qk\bar{r}$ units to compute the outputs of the "hard inputs". (Here we use some natural closure properties of the computational model). Thus, $Suc^{Res}_{kr^*+2qk\bar{r}}(f^{\oplus k}) \geq 1 - 2^{-\Omega(k)}$. •

Remark 2 *It should be noted that the function f constructed here isn't as pathological as it may seem at first glance. Impagliazzo's hard core theorem [6], shows that (at least in the boolean circuit model) every function f with $Suc^{Size}_s(f) \leq 1 - q$ has a large subset of the inputs on which any (slightly smaller) circuit succeeds with probability roughly $1/2$. In our example the "hard core" of f is the function g. The unnatural state of affairs in our example is that the function is easy outside of the hard core, and deciding whether an input is in the hard core is an easy computational task.*

3.2. Conclusions for specific models

In order to use the counterexample from the previous section we will show the existence of the required "building block" functions g and h for various computational models.

We will use the same function as "h" in all constructions. Namely we choose $q = 2^{-l}$ for integer l, and define $h : \{0,1\}^l \to \{0,1\}$ to take the value one if all its inputs are zeroes, and zero otherwise. It is immediate to verify that h is in $Comm_l, Depth_l, Size_{O(l)}$, and accepts a q-fraction of its inputs.

3.2.1 Communication Protocols

For communication complexity we use the inner product function as "g". More precisely, consider the function $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined in the following way: $g(x,y) = \sum_{1 \leq i \leq n} x_i y_i mod 2$. As all functions over n bit inputs, $g \in Comm_n$. It is known that g is very hard on average given $n/4$ bits of communication[5].

Theorem 1 *[2]* $Adv^{Comm}_{n/4}(g) \leq 2^{-\Omega(n)}$.

Using lemma 1 we conclude that $Adv^{Comm}_{\Omega(kc)}(f^{\oplus k}) > Adv^{Comm}_c(f)$ for a sufficiently small constant q and a sufficiently large constant k. Thus, there are no strong direct product assertions for communication protocols.

Remark 3 *In this example c is very large and in particular $\bar{c} = n$ this can be avoided by "padding" the inputs of the two players to increase n.*

[5]Preparing for section 4, we remark that the proof of that statement works by showing that g has very low discrepancy.

3.2.2 Decision Trees

For decision trees we choose g to be the parity function $g(x) = \oplus_{1 \leq i \leq n} x_i$. It is immediate that $g \in Depth_n$ and $Adv^{Depth}_{n-1}(g) = 0$. Once again we can use lemma 1 to get a counterexample with the same behavior of parameters as the counterexample for communication protocols.

3.2.3 Boolean Circuits

The counterexample applies also to boolean circuits. That is, it is possible to prove the existence of a function g which is hard on average for some size and easy for slightly larger size. (Any function can be computed in size roughly $2^n/n$ and a counting argument could be used to show that there is a function which is hard on average for slightly smaller size). However, in this model, there are much stronger counterexamples. This is shown by the following "mass production" theorem by Uhlig.

Theorem 2 *[18] For every function $f : \{0,1\}^n \to \{0,1\}$ and $k = 2^{o(n/\log n)}$ $f^{\oplus k} \in Size_{2^n n^{-1} + o(2^n n^{-1})}$.*

This means that if f has near maximal circuit complexity then huge savings can be made when computing f on many inputs. This is much stronger than our counterexample as in this setup savings can actually be made on the worst $x_1, \cdots, x_k$ whereas our example only shows that savings can be made on "average" $x_1, \cdots, x_k$.

It should be pointed out that this is not the case in decision trees. It is easy to show that if $f \notin Depth_d$ then $f^{\oplus k} \notin Depth_{kd}$, and no savings can be made on the worst case. It is not known whether significant savings can be made on the worst case in communication protocols. This question is known as the "direct sum conjecture" of Karchmer, Raz and Wigderson, [10]. In some sense, (see [10]) a negative answer will supply a lower bound for boolean circuits with small depth.

3.3. How bad is this counterexample?

It seems that the counterexample is not so bad in the sense that it does not contradict our intuition as to why direct product assertions are true. When proving a direct product assertion, the main task is to show that the algorithm does not benefit from correlating computations on different inputs. In the counterexample we presented no real correlations occur. Instead, the algorithm uses its resource in an unbalanced way, spending a lot of it on particular inputs. In particular, the algorithm is able to compute f on single instances with advantage greater than p. This is something which is ruled out when $r' \leq r$.

We would like to change the formulation of the direct product problem to rule out such cases. In the next two sections we suggest two such approaches. One involves adding assumptions on the function f, in hope that such assumptions can prevent the situation of the counterexample. Intuitively, if f is hard in a "robust way", any additional resources spent on f on one instance will result in a "loss" in another instance, and it will not be beneficial to treat the inputs unfairly. The other approach involves restricting the algorithm in a way that insures that it cannot have advantage greater than p when attempting to compute f on any single coordinate.

4. A discrepancy product theorem

In the previous section we've seen that a direct product assertion for communication protocols is not true if we allow the protocol trying to compute $f^{\oplus k}$ to pass slightly more bits than the protocol attempting to compute f. In this section we show that if f has "low discrepancy" then a strong direct product theorem holds for f.

A communication complexity problem $f(x,y)$ can be viewed as a matrix A such that $A_{xy} = f(x,y)$. It will be convenient to think of the outputs of a communication complexity problem as $\{-1,1\}$ rather than $\{0,1\}$, thus the matrix will have entries in $\{-1,1\}$. We assume the inputs of both players are of the same size which means that A is a square matrix. The choice of $\{-1,1\}$ is made so that the tensor product of A with itself k times, (denoted by $A^{\otimes k}$) is exactly the matrix of the communication problem $f^{\oplus k}$.

For a set $C \subseteq [N]$, we use χ_C to denote the *characteristic vector* of C, that is $(\chi_C)_i = 1$ for $i \in C$ and $(\chi_C)_i = 0$ for $i \notin C$.

Definition 2 *Let A be an N by N matrix with entries in $\{-1,1\}$. For a rectangle $R = C \times D$ where $C, D \subseteq [N]$ we define:*

$$disc_R(A) = \frac{|\chi_C^t A \chi_D|}{N^2}$$

The discrepancy of A is defined in the following way:

$$disc(A) = max_R disc_R(A)$$

where the maximum is taken over all rectangles $R = C \times D$ with $C, D \subseteq [N]$.

For a fixed rectangle $R = C \times D$, $\frac{|\chi_C^t A \chi_D|}{|C \times D|}$ measures how unbalanced is the matrix A in the rectangle R. If R is a rectangle reached in a leaf of a communication protocol then this quantity is the advantage the protocol gets in the rectangle R. The definition of $disc_R(A)$ multiplies this quantity by $\frac{|C \times D|}{N^2}$ to take into account the volume of the rectangle. More precisely, the advantage is multiplied by the volume of the rectangle R to give the contribution of R to the overall advantage of the protocol. This normalization is made so that low discrepancy will imply that the problem is hard on average. This is made formal in the following lemma.

Lemma 2 $Adv_c^{Comm}(A) \leq disc(A) 2^c$

Proof: (of lemma 2) Let P be the c-bit communication protocol which achieves $Adv_c^{Comm}(A)$. A c-bit communication protocol partitions A into 2^c disjoint rectangles. On each rectangle $R_i = C_i \times D_i$ the advantage of the protocol in the rectangle is given by $\frac{|\chi_{C_i}^t A \chi_{D_i}|}{|C_i \times D_i|}$. We can now bound the advantage of P:

$$Adv_c^{Comm}(A) = \sum_{1 \leq i \leq 2^c} \frac{|C_i \times D_i|}{N^2} \cdot \frac{|\chi_{C_i}^t A \chi_{D_i}|}{|C_i \times D_i|}) \leq 2^c disc(A)$$

•

The requirement that $disc(A)$ is small is stronger than that A is hard on average. Still, the most common way of showing that communication problems are hard on average is by showing that they have low discrepancy.

Remark 4 *In remark 2 we pointed out that the function of the counterexample has a large set of easy inputs, and it is possible to check whether a given input is easy. Intuitively, low discrepancy exactly avoids this kind of scenario as it enforces that in any large rectangle the function is hard. It is impossible that the two players pass few bits and reach a large set on which the function is easy.*

The main theorem of this paper shows that the discrepancy of a taking the tensor product of A with itself k times goes down exponentially with k.

Theorem 3 $disc(A^{\otimes k}) = O(disc(A))^{\frac{k}{3}}$

This has the following interpretation: Suppose, (as is often the case), that the fact that $Adv_c^{Comm}(A) \leq p$ follows from the fact that $disc(A) \leq p 2^{-c}$. In that case:

$$Adv_{kc/3}^{Comm}(A^{\otimes k}) \leq disc(A^{\otimes k}) 2^{kc/3}$$

$$\leq O(2^c disc(A))^{k/3} \leq O(p)^{k/3}$$

In words, we get a strong direct product theorem for A. This is stated with more generality in the next corollary.

Corollary 1 *For every c', $Adv_{c'}^{Comm}(A^{\otimes k}) \leq O(disc(A))^{k/3} 2^{c'}$.*

Remark 5 *While we do not know whether theorem 3 is tight, it is impossible to get that:* $disc(A^{\otimes k}) = disc(A)^{\frac{k}{3}}$ *Consider the following* N *by* N *matrix for* $N = 2^n$, $B_{ij} = (-1)^{i+j}$, *which corresponds to the following problem: Each of the two players gets an* n *bit vector and the players want to compute the exclusive or of the* $2n$ *bits. It is easy to see that* $disc(B) = 1/4$ *and does not depend on* N. *(To show that* $disc(B) \geq 1/4$ *note that* B *has a trivial* 2*-bit communication protocol in which each player sends the exclusive or of its bits). However,* $B^{\otimes k}$ *is equal to the matrix* B *of size* $N^k \times N^k$. *Thus,* $disc(B^{\otimes k}) = disc(B)$ *for all* k.

The proof of the theorem will require the definition of the spectral norm of a matrix A.

Definition 3 *For a vector* x *we use* $||x||_2$ *to denote the L2-norm of* x. *For a matrix* A, $||A||_2$ *is defined to be* $max_{x:||x||_2=1}||Ax||_2$.

It will be useful to consider equivalent definitions of this norm.

Fact 1 *Equivalent definitions for* $||A||_2$ *are:*

1. $||A||_2 = max_{x:||x||_2=1, y:||y||_2=1}|x^tAy|$.
2. $||A||_2 = max\{\sqrt{\lambda}|\ \lambda$ *is an eigenvalue of* $A^tA\}$

A useful property of $||A||_2$ is that it is multiplicative under tensor product.

Fact 2 $||A^{\otimes k}||_2 = ||A||_2^k$

Proof: (of fact 2) The proof of fact 2 consists of two steps: The first is to show this is true for symmetric matrices. If A is symmetric then there is an orthonormal basis of eigenvectors of A and $||A||_2 = |\lambda|$, where λ is the maximal eigenvalue of A in absolute value. It is easy to see that the eigenvalues of $A^{\otimes k}$ are exactly all products of eigenvalues of A. Thus, $||A||_2 = |\lambda|^k$. The fact now follows for non symmetric matrices by using the second item of fact 1, and the observation that $(A^{\otimes k})^tA^{\otimes k} = (A^tA)^{\otimes k}$. •

In the remainder of this section we prove theorem 3. The first step is to express the discrepancy in terms of the spectral norm. We then use the multiplicativity of the spectral norm to get the conclusion. The second item of fact 1 enables us to upper bound the discrepancy using the spectral norm.

Lemma 3 $disc(A) \leq \frac{||A||_2}{N}$

Proof: (of lemma 3) Let $R = C \times D$ be the rectangle such that $disc(A) = disc_R(A)$. We have that $disc(A) = \frac{|\chi_C^tA\chi_D|}{N^2}$. We define $x = (\chi_C)/\sqrt{|C|}$ and $y = (\chi_D)/\sqrt{|D|}$. Note that $||\chi_C||_2 = ||\chi_D||_2 = 1$. it follows from the first item of fact 1 that

$$||A||_2 \geq |x^tAy| = \frac{|\chi_C^tA\chi_D|}{\sqrt{|C|}\sqrt{|D|}} \geq \frac{|\chi_C^tA\chi_D|}{N} = disc(A)N$$

•

In [14], Nisan and Wigderson address the so called "log-rank conjecture" and show that $disc(A) = \Omega(1/rank(A)^{3/2})$. We will use ideas from that paper to lower bound the discrepancy using the spectral norm.

Lemma 4 $disc(A) = \Omega\left(\frac{||A||_2}{N}\right)^3$

We start by showing that theorem 3 easily follows from lemma 4.

Proof: (of theorem 3)

$$disc(A^{\otimes k}) \leq \frac{||A^{\otimes k}||_2}{N^k} = \left(\frac{||A||_2}{N}\right)^k = O\left(disc(A)\right)^{k/3}$$

The first inequality follow from lemma 3. The second follows from fact 2, and the third follows from lemma 4. •

We want to prove lemma 4 in a similar way to the previous lemma. The first step is a way to transform a bilinear form with arbitrary vectors into one with characteristic vectors. Such a transformation was given by Nisan and Wigderson in [14].

Lemma 5 *[14] Let* u, v *be vectors such that* $||u||_\infty, ||v||_\infty \leq 1$, *then there exist a rectangle* $R = C \times D$ *such that* $|\chi_C^tA\chi_D| \geq |u^tAv|/4$.

For completeness we give the proof of this lemma.

Proof: (of lemma 5) We "split" u and v to their positive coordinates and negative coordinates, having $u = u^+ - u^-$ and $v = v^+ - v^-$. We now have that $||u^+||_\infty, ||u^-||_\infty, ||v^+||_\infty, ||v^-||_\infty \leq 1$ and the four vectors are non-negative. Since $|u^tAv| = |(u^+)^tA(v^+) - (u^+)^tA(v^-) - (u^-)^tA(v^+) + (u^-)^tA(v^-)|$ then one of the four terms is larger in absolute value than $|u^tAv|/4$. Setting C to be the non-negative coordinates of the u-part of this term and D to be the non-negative coordinates of the v-part of this term, we have that $|\chi_C^tA\chi_D| \geq |u^tAv|/4$. •

We are now tempted to use lemma 5 directly to prove lemma 4. That is start from u, v such that $||A||_2 = u^tAv$ and get a rectangle $R = C \times D$ with roughly the same value. This will not do since to get a bound on $disc(A)$ we have to divide by N^2. Thus, the above argument only gives the non-impressive estimate: $disc(A) \geq \frac{||A||_2}{4N^2}$. Had it been the case that u and v had $||u||_\infty, ||v||_\infty \leq \rho < 1$ we could deduce that $\rho^2 disc(A) \geq \frac{||A||_2}{4N^2}$ and do better. We will now show that $||A||_2$ is obtained by such u and v.

Lemma 6 *There are vectors u, v with $||u||_\infty, ||v||_\infty \leq \rho$ such that $|u^t Av| \geq ||A||_2 - \frac{2\sqrt{N}}{\rho}$*

Proof: (of lemma 6) Let x and y be vectors such that $||x||_2 = ||y||_2 = 1$ and $||A||_2 = |x^t Ay|$. Let $I = \{i | x_i > \rho\}$ and $J = \{j | y_j > \rho\}$. Let u be the vector obtained from x by setting the coordinates in I to zero, and v be the vector obtained from y by setting the coordinates in J to zero. Note that $|I|, |J| \leq 1/\rho^2$, (since otherwise the contribution of elements in I (J) to the norm of x (y) is greater than one). We now have that:

$$|u^t Av| \geq ||A||_2 - |\sum_{i \in I, j \in [N]} x_i a_{ij} y_j + \sum_{i \in [N], j \in J} x_i a_{ij} y_j|$$

We will now argue that the two terms on the right hand side are small because they involve small rectangles. We will bound the first term, and the second can be bounded the same way.

$$|\sum_{i \in I, j \in [N]} x_i a_{ij} y_j| \leq \sum_{i \in I, j \in [N]} |x_i y_j|$$

$$= \sum_{i \in I} |x_i| \sum_{j \in [N]} |y_j| \leq \sqrt{|I|}\sqrt{N} ||x||_2 ||y||_2 \leq \frac{\sqrt{N}}{\rho}$$

(where the second inequality follows from the Cauchy-Schwartz inequality). Plugging this in the previous calculation we have that:

$$|u^t Av| \geq ||A||_2 - \frac{2\sqrt{N}}{\rho}$$

•

Lemma 4 follows from the previous lemmas.

Proof: (of lemma 4) Given a matrix A, we set:

$$\rho = \frac{3\sqrt{N}}{||A||_2}$$

Let u, v be the vectors which existence is given by lemma 6. We now define new vectors $\bar{u} = u/\rho$ and $\bar{v} = v/\rho$. Note that $||\bar{u}||_\infty, ||\bar{v}||_\infty \leq 1$. Using lemma 5 there exists a rectangle $R = C \times D$ such that

$$|\chi_C^t A \chi_D| \geq \frac{|\bar{u}^t A \bar{v}|}{4} = \frac{|u^t Av|}{4\rho^2} = \frac{||A||_2 - \frac{2\sqrt{N}}{\rho}}{4\rho^2} = \frac{||A||_2^3}{108N}$$

(Note that this result is valid even if $\rho > 1$. In that case, it follows directly without using anything). We conclude that $disc(A) = \Omega\left(\frac{||A||_2}{N}\right)^3$. •

5. Fair decision trees

In this section we prove an optimal direct product theorem for fair decision trees. The proof is quite simple and mimics the technique of [13]. The purpose of this section is to promote the notion of "fairness" which can perhaps be extended to more interesting models of computation. We start with the definition of "fairness" for decision trees.

Definition 4 *A decision tree over inputs $x_1, \cdots, x_k$ is* $(d_1, \cdots, d_k)$-fair *if for every $1 \leq i \leq k$ and on every path from the root to a leaf the decision tree queries at most d_i bits from x_i. A decision tree is d-fair if it is $(d, \cdots, d)$-fair.*

Let $FairDepth_{d_1,\cdots,d_k}$ denote the class of functions over inputs $x_1, \cdots, x_k$ computed by $d_1, \cdots, d_k$ decision trees. Let $FairDepth_{kd}$ denote $FairDepth_{d,\cdots,d}$.

Theorem 4 $Adv_{kd}^{FairDepth}(f^{\oplus k}) \leq Adv_d^{Depth}(f)^k$.

The proof of this theorem is by induction and is almost similar to that of [13]. To simplify the notation we will prove it for $k = 2$. The proof for general k follows the same way. To have a stronger induction hypothesis we will prove the following stronger version.

Lemma 7 *For every two functions f_1, f_2 and numbers d_1, d_2*

$$Adv_{d_1,d_2}^{FairDepth}(f \oplus g) \leq Adv_{d_1}^{Depth}(f_1) \cdot Adv_{d_2}^{Depth}(f_2)$$

Proof: (of lemma 7) We prove the lemma by induction over $d_1 + d_2$. If $d_1 + d_2 = 0$ then the decision tree does not base its answer on the inputs. It is standard to check that indeed $Adv_{0,0}^{FairDepth}(f_1 \oplus f_2) = Adv_0^{Depth}(f_1) \cdot Adv_0^{Depth}(f_2)$. (From the point of view of a tree which makes no queries the outputs of f_1 and f_2 are *independent* bit variables). To bound $Adv_{d_1,d_2}^{FairDepth}(f_1 \oplus f_2)$ for $d_1 + d_2 > 0$, let T be a decision tree which achieves this advantage. Without loss of generality the first query of T is from x_1. We will use the notation $x_1 = (y, b)$ where b is the bit queried by T and y is the remaining bits. We denote the two sub-trees of T by T^0 and T^1 respectively. For $b \in \{0, 1\}$, we define functions $g^b(y) = f_1((y, b))$. We now have:

$$Adv_{d_1,d_2}^{FairDepth}(f_1 \oplus f_2) = \sum_{b \in \{0,1\}} (1/2) \cdot Adv^{T_b}(g^b \oplus f_2)$$

Here, $Adv^T(f)$ is used to denote the advantage of T on f. However, T^0 and T^1 are of depth $d_1 - 1 + d_2$, and are in

$FairDepth_{d_1-1,d_2}$. Applying our induction hypothesis we continue and get that:

$$\leq \sum_{b\in\{0,1\}} (1/2) \cdot Adv^{FairDepth}_{d_1-1,d_2}(g^b \oplus f_2)$$

$$\leq Adv^{Depth}_{d_2}(f_2) \sum_{b\in\{0,1\}} (1/2) \cdot Adv^{Depth}_{d_1-1}(g^b)$$

Consider trees P^0, P^1 which achieve the advantage on g^0, g^1. We now construct a tree P of depth d_1 which starts by querying b and depending on the outcome activates P^b. We now have that:

$$\sum_{b\in\{0,1\}} (1/2) \cdot Adv^{Depth}_{d_1-1}(g^b) = Adv^p(f_1) \leq Adv^{Depth}_{d_1}(f_1)$$

The first equality follows from the definition of P and the second from the fact that P is of depth d_1. Plugging this in our previous calculation we prove the lemma. •

6. Discussion and open problems

There are two natural lines of open problems which relate to the two approaches suggested in this paper for proving strong direct product assertions.

The first line involves coming up with the right restrictions on functions which enable proving strong direct product assertions in other models of computation. There are also more concrete problems regarding discrepancy and communication games. For example can the notion of discrepancy be altered to match other probability distributions on the inputs of the two players, (not just the uniform distribution) while still enabling a strong direct product assertion. Another problem is whether the approach of this paper can be extended to one-sided discrepancy, which removes the absolute value in definition 2. These two problems are motivated by Razborov's proof [17] of the lower bound to the disjointness function [1, 9] which uses one-sided discrepancy in a non-uniform distribution.

The second line involves extending the "fairness" notion suggested in this paper to other computational models. A natural place to start is communication games or even one round communication games. This brings us to the issue of generalizing the definition of fair decision trees to other models of computation.

6.1. Fairness for general models of computation

While it is easy to define fairness for decision trees, how does one define a fair communication protocol or a fair computation in general? Before making our suggestions, let us examine what is the role of fairness in the argument. Recall the information theoretic analogue presented in the beginning of the introduction. The goal there is to predict the outcome of an exclusive-or of k independent biased coins. The direct product question is the analogue of this question in the sense that from of the point of view of any algorithm in Res_r the function, (or more precisely the exclusive or of the outcome of the function with the answer of the algorithm) is a biased coin. What makes the computational version difficult is that the algorithm may make these biased coins correlated. However, we still want to have that each individual coin is not fully determined. In other words, that the algorithm attempting to compute $f^{\oplus k}$ cannot compute any of the individual outcomes of $f(x_i)$. When $r' \leq r$, (as is the case in Yao's XOR-lemma) this happens simply because the protocol attempting to compute $f^{\oplus k}$ is "small" enough to be bounded by the hypothesis. The role of the fairness restriction is to insure this situation when $r' > r$. Having this in mind we can now suggest two approaches to impose a fairness restriction on general models of computation.

6.1.1 A syntactic approach

We want that a protocol using $r' > r$ units of the resource will not be able to compute any individual outcome of $(f(x_1), \cdots, f(x_k))$ too well. In fair decision trees this is guaranteed because once you fix $k-1$ inputs of a d-fair decision tree, the induced tree is of depth d, and is thus bounded by the hypothesis of the direct product assertion. This leads us to the following definition of syntactic fairness for a general model of computation.

Definition 5 *An algorithm $P' \in Res_{r'}$ with inputs $x_1, \cdots, x_k$ is r-fair if for every $1 \leq i \leq k$ and every fixing $a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_k$ for the variables $x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_k$ $P'(a_1, \cdots, a_{i-1}, x, a_{i+1}, \cdots, a_k)$ can be simulated by an algorithm $P(x)$ in Res_r*

It may very well be that syntactic fairness is a very strong restriction. However, at this point we do not know whether syntactic fairness gives a strong direct product assertion in communication games.

6.1.2 A semantic approach

The role of syntactic fairness is to guarantee that the algorithm cannot compute any individual outcome too well. It is natural to replace it by the more semantic condition it is supposed to entail. We need to know what the algorithm attempting to compute $f^{\oplus k}$ "thinks" about individual values of $f(x_i)$. Unfortunately, we only get to know this in the concatenation variant of the direct product problem defined in remark 1. In this setup the algorithm is supposed to compute $f^{(k)}(x_1, \cdots, x_k) = (f(x_1), \cdots, f(x_k))$. This leads to the following definition:

Definition 6 *An algorithm $P' \in Res_{r'}$ with inputs $x_1, \cdots, x_k$ and outputs in $\{0,1\}^k$ is (r,p)-semantically fair for f if for every $1 \leq i \leq k$, $\Pr_{x_1,\cdots,x_k}[P_i(x_1, \cdots, x_k) = f(x_i)] \leq p$.*

Syntactic fairness automatically implies semantic fairness. However, semantic fairness is far less restricting and may be sufficient to prove strong direct product assertions in the concatenation variant.

7. Acknowledgments

I would like to thank my advisor Avi Wigderson for introducing me to this area and for his endless guidance and support. I would like to thank Russell Impagliazzo and Salil Vadhan for helpful discussions. I am also thankful to anonymous referees for pointing out some relevant previous work and helpful comments.

References

[1] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science*, pages 337–347, Toronto, Ontario, Canada, 27–29 Oct. 1986. IEEE.

[2] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, Apr. 1988.

[3] U. Feige and O. Verbitsky. Error reduction by parallel repetition—a negative result (preliminary version). In *Proceedings, Eleventh Annual IEEE Conference on Computational Complexity*, pages 70–76, Philadelphia, Pennsylvania, 24–27 May 1996. IEEE Computer Society Press.

[4] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[5] O. Goldreich, N. Nisan, and A. Wigderson. On yao's xor-lemma. Technical report, Electronic Colloquium on Computational Complexity, 1995. `http://www.eccc.uni-trier.de/eccc`.

[6] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, Milwaukee, Wisconsin, 23–25 Oct. 1995. IEEE.

[7] R. Impagliazzo, R. Raz, and A. Wigderson. A direct product theorem. In *9th Structure in Complexity Theory Conference*, pages 88–96, 1994.

[8] 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.

[9] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection (preliminary version). In *Proceedings, Structure in Complexity Theory, Second Annual Conference*, pages 41–47, Cornell University, Ithca, NY, 16–19 June 1987. IEEE Computer Society Press.

[10] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via direct sum in communication complexity. In *6th Structure in Complexity Theory Conference*, pages 299–304, 1991.

[11] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge university press, 1997.

[12] L. A. Levin. One-way functions and pseudorandom generators. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 363–365, Providence, Rhode Island, 6–8 May 1985.

[13] N. Nisan, S. Rudich, and M. Saks. Products and help bits in decision trees. In *35th Annual Symposium on Foundations of Computer Science*, pages 318–329, Santa Fe, New Mexico, 20–22 Nov. 1994. IEEE.

[14] N. Nisan and A. Wigderson. On rank vs. communication complexity. In *35th Annual Symposium on Foundations of Computer Science*, pages 831–836, Santa Fe, New Mexico, 20–22 Nov. 1994. IEEE.

[15] I. Parnafes, R. Raz, and A. Wigderson. Direct product results and the GCD problem, in old and new communication models. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 363–372, El Paso, Texas, 4–6 May 1997.

[16] R. Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, June 1998.

[17] A. A. Razborov. On the distributed complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 14 Dec. 1992. Note.

[18] D. Uhlig. On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Math. Notes Acad. Sci. USSR 15*, pages 558–562, 1974.

[19] 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.