comput. complex. **21** (2012), 197 – 244

1016-3328/12/020197-48
published online June 2, 2012
DOI 10.1007/s00037-012-0043-7

computational complexity

IMPROVED DIRECT PRODUCT THEOREMS FOR RANDOMIZED QUERY COMPLEXITY

Andrew Drucker

Abstract. The "direct product problem" is a fundamental question in complexity theory which seeks to understand how the difficulty in computing a function on each of k independent inputs scales with k. We prove the following direct product theorem (DPT) for query complexity: if every T-query algorithm has success probability at most $1-\varepsilon$ in computing the Boolean function f on input distribution μ, then for $\alpha \leq 1$, every $\alpha\varepsilon Tk$-query algorithm has success probability at most $(2^{\alpha\varepsilon}(1-\varepsilon))^k$ in computing the k-fold direct product $f^{\otimes k}$ correctly on k independent inputs from μ. In light of examples due to Shaltiel, this statement gives an essentially optimal trade-off between the query bound and the error probability. Using this DPT, we show that for an absolute constant $\alpha > 0$, the worst-case success probability of any $\alpha R_2(f)k$-query randomized algorithm for $f^{\otimes k}$ falls exponentially with k. The best previous statement of this type, due to Klauck, Špalek, and de Wolf, required a query bound of $O(bs(f)k)$.
Our proof technique involves defining and analyzing a collection of martingales associated with an algorithm attempting to solve $f^{\otimes k}$. Our method is quite general and yields a new XOR lemma and threshold DPT for the query model, as well as DPTs for the query complexity of learning tasks, search problems, and tasks involving interaction with dynamic entities. We also give a version of our DPT in which decision tree size is the resource of interest.

Keywords. Direct product theorems, query complexity, decision trees, average-case complexity, hardness amplification.

Subject classification. 68Q17.

1. Introduction

1.1. Direct product theorems. Suppose some Boolean function $f(x)$ on n input bits is "hard to compute" for a certain computational model. It seems that computing the k-tuple $f^{\otimes k}(x^1, \ldots, x^k) := (f(x^1), \ldots, f(x^k))$ on independent inputs $x^1, \ldots, x^k$ should be "even harder." The intuition is that the k tasks to be performed appear separate and unrelated and that with more tasks one is more likely to make a mistake. One way to make this idea more precise is to study the *direct product problem*, in which we try to prove statements of the following form:

Suppose every algorithm using resources at most T has success probability at most p in computing f. Then, every algorithm using resources at most T' has success probability at most p' in computing $f^{\otimes k}$ on k independent inputs to f.

Such a result is called a *direct product theorem (DPT)*. The direct product problem can be contrasted with a second, related question, the *direct sum problem*, which studies how the complexity of solving k instances of a problem scales with k, when we are only interested in algorithms that succeed with high probability (or probability 1). For a recent overview of the direct sum problem in query complexity, and proofs of some new results, see Jain *et al.* (2010).

Depending on the computational model and our interests, T and T' might measure time, communication, or any other resource. The success probability could be with respect to some input distribution μ, in which case it is natural to assume in the k-fold setting that the inputs are drawn independently from μ; we call this the *average-case* setting. However, one can also consider the case where p is a bound on the *worst-case* success probability of a randomized algorithm, ranging over all inputs to f; we then try to establish an upper-bound p' on the worst-case success probability of query-bounded algorithms for $f^{\otimes k}$. The strength of a direct product theorem can be measured in terms of the dependence of the parameters T', p' on T, p, k, and, possibly, on the function f itself. We want T' to be large and p' to be small to establish that the k-fold problem is indeed "very hard."

There is also an important variant of the direct product problem, in which we are interested in computing the "k-fold XOR" $f^{\oplus k}(x^1, \ldots, x^k) := f(x^1) \oplus \cdots \oplus f(x^k)$ of k independent inputs to f. An *XOR lemma* is a result that upper-bounds the success probability p' achievable by algorithms for $f^{\oplus k}$ using T' resources, under the assumption that any algorithm using T resources has success probability at most p.[1] An obvious difference from DPTs is that in an XOR lemma, p' must always be at least $1/2$, since $f^{\oplus k}$ is Boolean and the algorithm could simply guess a random bit. The hope is that $(p' - 1/2)$ decays exponentially with k. Research on XOR lemmas has proceeded in parallel with research on direct product theorems; the known results are of similar strength (with some exceptions), and in some cases, there are reductions known from XOR lemmas to DPTs or vice versa; see Unger (2009); Impagliazzo & Kabanets (2010) for an overview and recent results of this type.

The direct product problem has been studied extensively in models such as Boolean circuits, for example, in Goldreich *et al.* (1995); Impagliazzo & Wigderson (1997); Impagliazzo *et al.* (2010); communication protocols (Impagliazzo *et al.* 1994; Shaltiel 2003; Klauck *et al.* 2007; Lee *et al.* 2008; Viola & Wigderson 2008); and query algorithms (Impagliazzo *et al.* 1994; Nisan *et al.* 1999; Shaltiel 2003; Klauck *et al.* 2007). In all of these models, an optimal T-bounded algorithm that attempts to compute f can always be applied independently to each of k inputs, using at most $T' = Tk$ resources and succeeding with probability $p' = p^k$, so these are the "ideal," strongest parameters one might hope for in a DPT. However, direct product statements of such strength are generally false, as was shown by Shaltiel (2003), who gave a family of counterexamples which applies to all "reasonable" computational models. We will describe these examples (specialized to the query model) in Section 4.[2]

[1] Terminology varies somewhat in the literature. For instance, what we call XOR lemmas are called "direct product theorems" in Shaltiel (2003), and what we refer to as direct product problems are called the "concatenation variant" by Shaltiel.

[2] Shaltiel calls a DPT "strong" if it applies to all p and T and its parameters satisfy $p' \leq p^{\Omega(k)}$ and $T' \geq \Omega(Tk)$. His counterexamples rule out strong DPTs for most computational models. In later works, the modifier "strong"

Thus, all DPTs shown have necessarily been weaker in one of several ways. First, researchers have restricted attention to algorithms of a special form. Shaltiel (2003) showed a DPT with the "ideal" parameters above holds for the query model, if the algorithm is required to query each of the k inputs exactly T times. He called such algorithms "fair."[3] A similar result for a special class of query algorithms called "decision forests" was shown earlier in Nisan *et al.* (1999).

Second, DPTs have been shown for unrestricted algorithms, but using resource bounds whose strength depends on properties of the function f. These results require the resource bound T' to scale as $\mathcal{D}(f)k$, where $\mathcal{D}(f)$ is some complexity measure that can be significantly smaller than the resources needed to compute a single instance of f. For example, Klauck *et al.* (2007) showed that for any f and any $\gamma > 0$, a DPT holds for f in which the achievable worst-case success probability p' is at most $(1/2 + \gamma)^k$, provided $T' \leq \alpha \cdot bs(f)k$ for some constant $\alpha = \alpha(\gamma) > 0$. Here, $bs(f)$ is the *block sensitivity* of f (Nisan 1991; Buhrman & de Wolf 2002), a complexity measure known to be related to the randomized query complexity by the inequalities $R_2(f)^{1/3} \leq bs(f) \leq R_2(f)$ (suppressing constant factors). Now, one can always compute f correctly on k instances with high probability using $O(R_2(f)k \log k)$ queries. For many functions, including random functions, $bs(f) = \Theta(R_2(f))$, so in these cases the DPT of Klauck *et al.* (2007) gives a fairly tight result. However, examples are known (Buhrman & de Wolf 2002) where $bs(f) = O(\sqrt{R_2(f)})$, so the number of queries allowed by this DPT can be significantly less than one might hope.

Klauck, Špalek, and de Wolf also proved DPTs for *quantum* query algorithms computing f, in which the worst-case success probability p' drops exponentially in k if the number of allowed quantum queries is $O(\sqrt{bs(f)k})$. For symmetric functions, direct

Footnote 2 continued
has been used in a somewhat broader way. We will not use this terminology in the present paper.

[3]Actually, Shaltiel proved, in our terms, an optimal XOR lemma for fair algorithms, but as he noted, this implies an optimal DPT, and his proof method can also be modified to directly prove an optimal DPT for fair algorithms.

product theorems of a strong form were proved for quantum query complexity by Ambainis *et al.* (2009). Špalek (2008) proved a DPT for quantum query algorithms where the resource bound T' scales in terms of a complexity measure called the *multiplicative quantum adversary*. Quite recently (after a preprint of our paper appeared), a sequence of works (Sherstov 2011; Ambainis *et al.* 2011; Lee & Roland 2011) dramatically advanced our understanding of the direct product problem in the quantum query model. This culminated in a DPT for quantum queries (Lee & Roland 2011) in which the success probability decays exponentially even as the query bound scales as $\Omega(Q_2(f)k)$. Here, $Q_2(f)$ is the bounded-error quantum query complexity of a (possibly non-Boolean) function f.

In the model of communication protocols, several types of results have been shown. DPTs have been given for specific functions: For example, in Klauck *et al.* (2007) a DPT was proved for the quantum communication complexity of the Disjointness function, and a classical analogue was proved by Klauck (2010). On the other hand, general DPTs have been given, whose resource bound scales in terms of complexity measures that may be significantly smaller than the communication complexity of f. For example, in communication complexity, DPTs have been shown whose strength is related to the so-called *discrepancy* of f (Shaltiel 2003; Lee *et al.* 2008).

Since the present work, there has been significant progress in the communication model. In the public-coin randomized setting, Jain (2011) showed a strong general-purpose DPT for one-way communication, and new DPTs were shown for two-way communication in (Jain 2011; Jain *et al.* 2012). Sherstov (2011) gave a new DPT for quantum communication, whose resource bound scales as $\Omega(\mathrm{GDM}(f)k)$, where $\mathrm{GDM}(f)$ is the lower bound on quantum communication complexity obtained by the *generalized discrepancy method*—the strongest lower-bound technique known in the quantum setting.

In the Boolean circuit model, despite intensive study, the known results are quantitatively much weaker and in particular require T' to *shrink* as k grows in order to make the success probability p' decay as k grows. It is at least known that, under this limitation,

a DPT with $p' = p^k$ can be shown using (Impagliazzo 1995; Holenstein 2005), as remarked in Impagliazzo & Kabanets (2010).

1.2. Our results. Our first result is the following direct product theorem in the average-case setting:

THEOREM 1.1. Suppose f is a Boolean function and μ is a distribution over inputs to f, such that any T-query randomized algorithm has success probability at most $(1-\varepsilon)$ in computing f on an input from μ. Then for $0 < \alpha \leq 1$, any randomized algorithm making $\alpha\varepsilon Tk$ queries has success probability at most $(2^{\alpha\varepsilon}(1-\varepsilon))^k < (1 - \varepsilon + .84\alpha\varepsilon)^k$ in computing $f^{\otimes k}$ correctly on k inputs drawn independently from μ.

We use Shaltiel's examples to show that the trade-off in Theorem 1.1 between the query bound and the error probability is essentially best-possible, at least for general functions f and for small values $\alpha < .01$. (For *specific* functions, the success probability will in some cases decay exponentially even when the number of queries allowed scales as Tk rather than εTk.) Theorem 1.1 reveals that small values of ε, as used in Shaltiel's examples, are the only major "obstruction" to strong, general direct product statements in the query model.

Using Theorem 1.1, we obtain the following DPT for worst-case error, which strengthens the worst-case DPT of Klauck *et al.* (2007) mentioned earlier:

THEOREM 1.2. For any Boolean function f and $0 < \gamma < 1/4$, any randomized algorithm making at most $\gamma^3 R_2(f)k/11$ queries has worst-case success probability less than $(1/2 + \gamma)^k$ in computing $f^{\otimes k}$ correctly.

It seems intuitive that some statement like Theorem 1.2 should hold, and proving such a DPT was arguably one of the major open problems in classical query complexity.[4]

[4]While classical query algorithms can be viewed as a subclass of quantum query algorithms, we note that Theorem 1.2 is incomparable to the more-recent quantum DPT proved in Lee & Roland (2011): Our result shows exponentially decaying success probability for a more restricted class of algorithms, but under a potentially larger query bound.

We also prove a new XOR lemma. Let $B_{k,p}$ denote the binomial distribution on k trials with success probability p.

THEOREM 1.3. Suppose that any T-query randomized algorithm has success probability at most $(1-\varepsilon)$ in computing the Boolean function f on an input from μ. Then for $0 < \alpha \leq 1$, any randomized algorithm making $\alpha\varepsilon Tk$ queries and attempting to compute $f^{\oplus k}$ on k inputs drawn independently from μ has success probability at most

$$\frac{1}{2}\left(1 + \Pr_{Y \sim B_{k,1-2\varepsilon}}[Y > (1-\alpha\varepsilon)k]\right),$$

which is less than $\frac{1}{2}\left(1 + [1 - 2\varepsilon + 6\alpha\ln(2/\alpha)\varepsilon]^k\right)$.

Compare the probability bound above with the success probability $\frac{1}{2}(1+(1-2\varepsilon)^k)$, which can be attained using Tk queries by attempting to solve each instance independently and outputting the parity of the guessed bits. The concrete estimate given in Theorem 1.3 is meant to illustrate how our bound approaches this value as $\alpha \to 0$. By a more careful use of Chernoff inequalities, one can get somewhat tighter bounds for specific ranges of α, ε. An XOR lemma for the worst-case setting can also be derived from our result.

In addition to our "ordinary" DPT (Theorem 1.1), we also prove a "threshold" DPT, which bounds the probability that a query-bounded algorithm for $f^{\otimes k}$ solves "many" of the k instances correctly. As one special case, we prove:

THEOREM 1.4. Let f be a (not necessarily Boolean) function such that any T-query algorithm has success probability at most $1-\varepsilon$ in computing f on an input from μ. Fix $\eta, \alpha \in (0,1]$. Consider any randomized algorithm $\mathcal{R}$ making at most $\alpha\varepsilon Tk$ queries on k independent inputs from μ. The probability that $\mathcal{R}$ computes f correctly on at least ηk of the inputs is at most

$$\Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (\eta - \alpha\varepsilon)k].$$

Using Chernoff inequalities, Theorem 1.4 gives success bounds that decay exponentially in k for any fixed $\alpha, \varepsilon, \eta$, provided $\eta > 1 - \varepsilon + \alpha\varepsilon$. As we will explain, Shaltiel's examples show that this cutoff is nearly best-possible. By setting $\eta := 1$ in Theorem 1.4, we also get an ordinary DPT for non-Boolean functions, which for typical parameter settings is stronger than the DPT we'd obtain by a straightforward generalization of our techniques for Theorem 1.1. This is the simplest way we know to get such a DPT.

Threshold DPTs have been proved for a variety of models, including, recently, for arbitrary Boolean functions in the quantum query model (Lee & Roland 2011). Unger (2009) showed how to derive threshold DPTs from XOR lemmas, and recent work of Impagliazzo & Kabanets (2010) gave a way to derive threshold DPTs from sufficiently strong DPTs; see also the earlier works cited in Unger (2009); Impagliazzo & Kabanets (2010). However, the results of Impagliazzo & Kabanets (2010) do not apply for our purposes, and the threshold DPT we prove is more general than we'd get by applying the results of Unger (2009) to our XOR lemma. In any case, the proof of our threshold DPT is, we feel, quite natural and actually forms the basis for the proof of our XOR lemma. Our method for proving threshold DPTs applies to very general threshold events: We give bounds on the probability that the set $S \subseteq [k]$ of instances solved correctly by a query-bounded algorithm is "large," in a sense specified by an arbitrary monotone collection $\mathcal{A}$ of subsets of $[k]$. Generalized threshold DPTs of this form were shown recently by Holenstein & Schoenebeck (2011) in the circuit model, for a rich class of computational tasks called "weakly verifiable puzzles"; as usual in the circuit model, these DPTs require T' to shrink with k. Our techniques appear unrelated to theirs.

We also prove new DPTs for relations (for which direct *sum* theorems were proved recently by Jain *et al.* (2010)), learning tasks, search problems, and errorless heuristics. Deterministic query algorithms can be equivalently viewed as *decision trees*, and we also prove a DPT for decision trees in which decision tree *size*, rather than depth (i.e., number of queries), is the resource of interest. Impagliazzo *et al.* (1994) gave a DPT for decision tree size with "ideal" success probability decay $p' = p^k$, but in the case where

the size is not allowed to scale with k, that is, the setting $T' = T$. By contrast, in our DPT, the success probability decays as $p^{\Omega(k)} = (1-\varepsilon)^{\Omega(k)}$, while the size bound T' scales as $T^{\Omega(\varepsilon k)}$.

Finally, we give a further generalization of our DPTs, in which the k objects being queried are *dynamic entities* rather than static strings—that is, the answers to current queries may depend on past queries. DPTs for dynamic interaction have been proved before (Maurer *et al.* 2007), but only for the case in which the number of queries to each entity is fixed in advance. (This is analogous to Shaltiel's result for "fair" algorithms.) We further discuss the relation to past work on dynamic interaction in Section 10.

In order to ease notation, in this paper we discuss only DPTs for total functions, but our results apply to partial functions, that is, functions with a restricted domain; the proofs are the same. Similarly, our theorems and proofs carry over without change to handle non-Boolean input alphabets, as well as heterogeneous query costs. Taken as a whole, our results provide a fairly complete picture of the "direct product phenomenon" for randomized query complexity, although there may still be room for improvement in some of our bounds. We hope this work may also help lead to a better understanding of the direct product problem in other, richer computational models.

1.3. Our methods. We first explain our method to prove our "basic" direct product theorem, Theorem 1.1. As mentioned earlier, Shaltiel (2003) proved an optimal DPT for "fair" decision trees, in which each of the k inputs receives T queries. Our proof method for Theorem 1.1 also yields an alternate proof of Shaltiel's result, and it is helpful to sketch how this works first. (Really, this "alternate proof" is little more than a rephrasing of Shaltiel's proof technique, but the rephrasing gives a useful perspective which helps us to prove our new results.)

Suppose that every T-query algorithm for computing f succeeds with probability at most $1-\varepsilon$ on an input from the distribution μ. Consider a fair Tk-query algorithm $\mathcal{D}$ for $f^{\otimes k}$, running on k independent inputs from μ. We think of the algorithm as a "gambler" who bets at k "tables," and we define a random variable $X_{j,t} \in [1/2, 1]$ that represents the gambler's "fortune" at the jth

table after $\mathcal{D}$ has made t queries overall to the k inputs. Roughly speaking, $X_{j,t}$ measures how well the algorithm is doing in determining the value of f on the jth input. When $\mathcal{D}$ queries the jth input, the jth fortune may rise or fall, according to the bit seen; we regard each bit revealed to be generated sequentially at random, conditioned on the bits queried so far. The fortunes are defined so that $X_{j,0} \leq 1 - \varepsilon$ for each j (reflecting the assumed hardness of f on μ), and so that no action by the algorithm leads to an expected gain in fortune.[5] It follows that $\mathbb{E}[\prod_{j \in [k]} X_{j,Tk}] \leq (1 - \varepsilon)^k$. But the fortunes are defined, so that $\mathbb{E}[\prod_{j \in [k]} X_{j,Tk}]$ upper-bounds the success probability of $\mathcal{D}$ in computing $f^{\otimes k}$. This gives the DPT for fair algorithms. A key fact underlying the success of this proof strategy is that after conditioning on any initial sequence of outcomes to the first $t \leq T$ queries by the algorithm, the k inputs remain independent.

If $\mathcal{D}$ is no longer required to be fair, but instead makes at most $\alpha\varepsilon Tk$ queries, then the individual fortune $X_{j,t}$ we define no longer has the same intuitive meaning after the jth input has been queried more than T times. (In this event, we simply set $X_{j,t}$ to $1/2$, so that the gambler cannot hope to increase the jth fortune.) However, the success probability of $\mathcal{D}$ can still be upper-bounded by $\mathbb{E}[\prod_{j \in S} X_{j,\alpha\varepsilon Tk}]$, where S is the (random) set of inputs that receive at most T queries. Counting tells us that fewer than $\alpha\varepsilon k$ of the inputs can lie outside of S, and each fortune is always at least $1/2$, so the success probability is at most $2^{\alpha\varepsilon k}\mathbb{E}[\prod_{j \in [k]} X_{j,\alpha\varepsilon Tk}] \leq 2^{\alpha\varepsilon k}(1 - \varepsilon)^k$, giving the statement of Theorem 1.1.

Our worst-case DPT for Boolean functions follows straightforwardly from Theorem 1.1, by an application of Yao's minimax principle. Our DPT for decision tree size requires a somewhat different analysis, in which we track the "size-usage" of each of the k inputs rather than their number of queries, but the basic approach is the same as in Theorem 1.1. In generalizing our method to prove our other results, however, we face a new wrinkle: The natural definitions of the "fortunes" $X_{j,t}$ in these settings are no longer bounded from below by $1/2$. For example, if $f : \{0,1\}^n \to B$, then

[5] In standard probabilistic terms, each individual sequence $X_{j,0}, X_{j,1}, \ldots$ is a *supermartingale*. We will not use this terminology in the paper.

we have $X_{j,t} \geq |B|^{-1}$, and a straightforward modification of the method described above gives a DPT whose strength degrades as $|B|$ grows. In other settings (e.g., the k-fold XOR setting), we will only have $X_{j,t} \geq 0$, and the method fails completely.[6]

To overcome this difficulty, we adopt a more general perspective. Our previous proof hinged on the fact that if a gambler plays neutral or unfavorable games at k tables with an initial (nontransferable) endowment of $1 - \varepsilon$ at each table, then the probability he reaches a fortune of 1 at every table is at most $(1 - \varepsilon)^k$. Note that this is just the success probability he would achieve if he followed an independent "all-or-nothing bet" strategy at each table. It is natural to wonder whether this strategy remains optimal if the gambler wants merely to reach a fortune of 1 at "sufficiently many" tables. Indeed, we prove (by an induction on the number of rounds of gambling) that this is true, where the meaning of "sufficiently many" can be specified by any monotone collection of subsets of $[k]$. Most of our generalizations of Theorem 1.1, as well as our XOR lemma, follow readily from this handy "gambling lemma," although care is required to define the correct fortunes in each case.

1.4. Organization of the paper. In Section 2, we review preliminaries that are used throughout the paper and that are needed to state and prove our "basic" DPTs, Theorems 1.1 and 1.2. We will introduce other definitions as needed in later sections. In Section 3 we prove Theorem 1.1, and in Section 4 we use Shaltiel's examples to analyze the tightness of this result. We prove Theorem 1.2 in Section 5.

In Section 6 we prove our "gambling lemma" (Lemma 6.1) and use it to prove a generalized threshold DPT for relations. Theorem 1.4 will follow as a special case. We also explain how our threshold DPT implies a DPT for the query complexity of certain learning tasks. We prove Theorem 1.3, our XOR lemma, in Section 7 (also using Lemma 6.1). We define search problems and errorless heuristics in Section 8 and give DPTs for these settings.

[6]One way to work around the problem is to simply add a small "buffer term" to the fortunes $X_{j,t}$. However, this leads to poorer bounds and does not yield our generalized threshold DPTs.

We prove our DPT for decision tree size in Section 9. In Section 10, we describe generalizations of our DPTs to settings involving interaction with dynamic entities. We end with some questions for future work.

2. Preliminaries

All of our random variables will be defined over finite probability spaces. Let $\mathrm{supp}(X)$ denote the support of a random variable X, that is, the set of values with nonzero probability. Let $\mu^{\otimes k}$ denote k independent copies of distribution μ.

2.1. Randomized decision trees and query complexity. A *decision tree* $\mathcal{D}$ over $\{0,1\}^n$ is a rooted, full binary tree (i.e., each node has either 0 or 2 children), in which interior vertices v are labeled by indices $\mathrm{ind}(v) \in [n]$ and leaf vertices are labeled by values $\ell(v)$ in some finite set B (often $B = \{0,1\}$). The *height* of $\mathcal{D}$ is the length of the longest descending path in $\mathcal{D}$. $\mathcal{D}$ defines a function $f_{\mathcal{D}} : \{0,1\}^n \to B$ in the following way. On input x we start at the root and follow a descending path through $\mathcal{D}$; at interior node v, we pass to the left subchild of v if $x_{ind(v)} = 0$, otherwise we pass to the right subchild of v. When we reach a leaf vertex v, we output the value $\ell(v)$. Any deterministic algorithm to compute f which queries at most t bits of x on any input can be modeled as a height-t decision tree, and we will freely refer to such a tree as a "t-query deterministic algorithm."

A *randomized decision tree* is a probability distribution $\mathcal{R}$ over deterministic decision trees. Upon receiving the input x, the algorithm samples $\mathcal{D} \sim \mathcal{R}$, then outputs $\mathcal{D}(x)$. (Every randomized query algorithm can be modeled in this fashion.) We write $\mathcal{R}(x)$ to denote the random variable giving the output of $\mathcal{R}$ on input x. We say that $\mathcal{R}$ is a *t-query randomized decision tree* if every decision tree in the support of $\mathcal{R}$ has height at most t.

For $\varepsilon \in [0,1]$ and a function f (not necessarily Boolean), we say that $\mathcal{R}$ *ε-computes* f if for all inputs x, $\Pr[\mathcal{R}(x) = f(x)] \geq 1 - \varepsilon$. Similarly, if μ is a distribution over inputs $x \in \{0,1\}^n$, we say that $\mathcal{R}$ *ε-computes f with respect to μ* if $\Pr_{x\sim\mu}[\mathcal{R}(x) = f(x)] \geq 1 - \varepsilon$, where the probability is taken over the random sample $x \sim \mu$ and the randomness used by $\mathcal{R}$.

For a function $f : \{0,1\}^n \to B$, we define $R_2(f)$, the *two-sided-error randomized query complexity of* f, as the minimum t for which there exists a t-query randomized decision tree which $1/3$-computes f. We define

$$\mathrm{Suc}_{T,\mu}(f) := 1 - \varepsilon,$$

where $\varepsilon \geq 0$ is the minimum value for which some T-query-bounded randomized algorithm $\mathcal{R}\varepsilon$-computes f with respect to μ. By standard arguments, this minimum exists and is attained by a deterministic height-T decision tree.

For $f : \{0,1\}^n \to B$ and $k \geq 1$, define $f^{\otimes k} : \{0,1\}^{kn} \to B^k$, the *$k$-fold direct product of f*, as $f^{\otimes k}(x^1, \ldots, x^k) := (f(x^1), \ldots, f(x^k))$. If f is Boolean, define the *k-fold XOR of f* as $f^{\oplus k}(x^1, \ldots, x^k) := f(x^1) \oplus \cdots \oplus f(x^k)$, where $\oplus$ denotes addition mod 2.

2.2. Binomial distributions and Chernoff bounds. Let $B_{k,p}$ denote the binomial distribution on k trials with bias p. That is, $B_{k,p}$ is distributed as $Y = \sum_{i=1}^k Y_i$, where the Y_i are independent and 0/1-valued with $\Pr[Y_i = 1] = p$. For $s \in \{0, 1, \ldots, k\}$, we have the explicit formula $\Pr[Y = s] = \binom{k}{s} p^s (1-p)^{k-s}$.

The following is a general form of Chernoff's inequality:

Lemma 2.1. (Dubhashi & Panconesi 2009, §1.3) Suppose $Y \sim B_{k,p}$, with $q := 1 - p$. Then for $t \in [0, q)$,

$$\Pr\left[Y > (p+t)k\right] \leq \left(\left(\frac{p}{p+t}\right)^{p+t} \left(\frac{q}{q-t}\right)^{q-t} \right)^k.$$

The following form of Chernoff's inequality will be more convenient for us.

Lemma 2.2. Let $\delta \in (0,1)$, and let $Y \sim B_{k,1-\delta}$. If $\beta \in (0, 1/2]$, then

$$\Pr[Y > (1 - \beta\delta)k] < \left[1 - \delta + 6\beta \ln(1/\beta)\delta\right]^k.$$

Proof. We apply Lemma 2.1 with $t := (1-\beta)\delta$; we find

$$\begin{aligned}
\Pr\left[Y > (1-\beta\delta)k\right] &= \Pr[Y > ((1-\delta) + (1-\beta)\delta)k] \\
&\leq \left(\left(\frac{1-\delta}{1-\beta\delta}\right)^{1-\beta\delta}\left(\frac{\delta}{\delta-(1-\beta)\delta}\right)^{\delta-(1-\beta)\delta}\right)^k \\
&= \left(\left(\frac{1-\delta}{1-\beta\delta}\right)^{1-\beta\delta}\beta^{-\beta\delta}\right)^k \\
&\leq \left((1-\delta+2\beta\delta)^{1-\beta\delta}\,\beta^{-\beta\delta}\right)^k, \qquad (2.3)
\end{aligned}$$

using $\beta\delta \leq 1/2$.

It is easy to verify that $(1-\delta+2\beta\delta) \geq \beta$, so that

$$(1-\delta+2\beta\delta)^{-\beta\delta}\cdot\beta^{-\beta\delta} \leq \beta^{-2\beta\delta} = e^{2\beta\ln(1/\beta)\delta}.$$

Now $2\beta\ln(1/\beta)\delta \leq 2/e < .74$. By convexity of e^x, we have $e^x \leq 1 + ((e^{.74}-1)/.74)\cdot x \leq 1 + 1.49x$ for all $x \in [0, .74]$. Thus, $e^{2\beta\ln(1/\beta)\delta} \leq 1+3\beta\ln(1/\beta)\delta$. Combining these facts with Eq. (2.3), we get

$$\begin{aligned}
\Pr\left[Y > (1-\beta\delta)k\right] &\leq \left[(1-\delta+2\beta\delta)(1+3\beta\ln(1/\beta)\delta)\right]^k \\
&< \left[1-\delta+6\beta\ln(1/\beta)\delta\right]^k. \qquad \square
\end{aligned}$$

The constant 6 in Lemma 2.2 is not best-possible. To apply the lemma, it is helpful to understand the behavior of the function $h(x) := x\ln(1/x)$. This function is increasing on $(0, e^{-1}]$, and as $x \to 0$, $h(x)$ approaches 0 only slightly more slowly than x itself: for an integer $n > 1$ we have

$$h\left(\frac{1}{2n\ln n}\right) = \frac{1}{2n\ln n}\cdot\ln(2n\ln n) = \frac{1}{n}\cdot\frac{\ln(2n\ln n)}{\ln(n^2)} < \frac{1}{n}.$$

3. Proof of Theorem 1.1

In this section, we prove our "basic" direct product theorem:

Theorem 1.1 (restated). Let f be a Boolean function for which $\mathrm{Suc}_{T,\mu}(f) \leq 1-\varepsilon$. Then for $0 < \alpha \leq 1$, $\mathrm{Suc}_{\alpha\varepsilon T k,\mu^{\otimes k}}(f^{\otimes k}) \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k < (1-\varepsilon+.84\alpha\varepsilon)^k$.

There is no requirement that T be an integer; this will be useful later in proving Theorem 1.2. The success bound $(2^{\alpha\varepsilon}(1-\varepsilon))^k$ above is actually valid for any $\alpha > 0$, but the bound is trivial whenever $\alpha \geq 2$, so we focus attention on a range where the bound is always meaningful.

Proof. The statement is trivial if $T = 0$ or $\varepsilon = 0$, so assume both are positive. By convexity, it is sufficient to show the statement for deterministic algorithms. Also, by a standard limiting argument, it is enough to prove this result under the assumption that $\mathrm{supp}(\mu) = \{0,1\}^n$; this ensures that conditioning on any sequence of query outcomes will be well-defined.

Next, we set up some notation and concepts relating to the computation of f on a single input; afterward, we will apply our work to the direct product setting.

For a string $u \in \{0,1,*\}^n$, let the distribution $\mu^{(u)}$ be defined as a sample from μ, conditioned on the event $[x_i = u_i, \forall i$ such that $u_i \in \{0,1\}]$. Let $|u|$ denote the number of 0/1 entries in u. Let $u[x_i \leftarrow b]$ denote the string u with the ith coordinate set to b. In our proof, we consider the bits of an input $\mathbf{y} \sim \mu$ to be generated sequentially at random as they are queried. Thus, if an input is drawn according to μ, and u describes the outcomes of queries made so far (with $*$ in the coordinates that have not been queried), we consider the input to be in the "state" $\mu^{(u)}$. If some index $i \in [n]$ is queried next, then the algorithm sees a 0 with probability $\Pr_{\mathbf{y}\sim\mu^{(u)}}[\mathbf{y}_i = 0]$, in which case the input enters state $\mu^{(u[x_i\leftarrow 0])}$; with the remaining probability, the algorithm sees a 1 and the input enters state $\mu^{(u[x_i\leftarrow 1])}$. Clearly, this interpretation is statistically equivalent regarding the input as being drawn from μ before the algorithm begins (this is the "principle of deferred decisions" of probability theory).

For each $u \in \{0,1,*\}^n$ with $|u| \leq T$, let

$$W(u) \;:=\; \mathrm{Suc}_{T-|u|,\mu^{(u)}}(f).$$

In words, $W(u)$ measures our "winning prospects" of computing f on μ, if we begin with a budget of T queries and our first $|u|$ queries reveal the bits described by u, and if we follow an optimal strategy thereafter. Clearly, $W(u) \in [1/2, 1]$, since an algorithm may simply guess a random bit. We make two more simple claims about this function.

LEMMA 3.1. 1. $W(*^n) \leq 1 - \varepsilon$.

2. For any $u \in \{0, 1, *\}^n$ with $|u| < T$, and any $i \in [n]$, $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W(u[x_i \leftarrow \mathbf{y}_i])] \leq W(u)$.

PROOF. 1. This is immediate from our initial assumption $\mathrm{Suc}_{T,\mu}(f) \leq 1 - \varepsilon$.

2. If the ith coordinate has already been queried (i.e., $u_i \in \{0, 1\}$), then $\mathbf{y}_i = u_i$ with probability 1, so $u[x_i \leftarrow \mathbf{y}_i] = u$ and the statement is trivial. So assume $u_i = *$. Let $\mathcal{R}_0$, $\mathcal{R}_1$ be algorithms making at most $T - (|u| + 1)$ queries and maximizing the success probabilities on $\mu^{(u[x_i \leftarrow 0])}$, $\mu^{(u[x_i \leftarrow 1])}$, respectively. Thus, the success probability of $\mathcal{R}_b$ is $W(u[x_i \leftarrow b])$. Consider an algorithm $\mathcal{R}$ that queries x_i, then runs $\mathcal{R}_b$ if the bit seen is b. $\mathcal{R}$ makes at most $T - |u|$ queries, and the success probability of $\mathcal{R}$ is $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W(u[x_i \leftarrow \mathbf{y}_i])]$. Thus, $W(u)$ is at least this value. □

Now we prove the Theorem. Let $\mathcal{D}$ be any deterministic algorithm making at most $M := \lfloor \alpha \varepsilon T k \rfloor$ queries, and attempting to compute $f^{\otimes k}$ on input strings $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$. For $j \in [k]$ and $0 \leq t \leq M$, let $u_t^j \in \{0, 1, *\}^n$ be the random string giving the outcomes of all queries made to $\mathbf{x}^j$ after $\mathcal{D}$ has made t queries (to the entire input). We need the following simple but important observation:

LEMMA 3.2. Condition on any execution of $\mathcal{D}$ for the first $t \geq 0$ steps, with query outcomes given by $u_t^1, \ldots, u_t^k$. Then the input is in the state $\mu^{(u_t^1)} \times \cdots \times \mu^{(u_t^k)}$. That is, the k inputs are independent, with $\mathbf{x}^j$ distributed as $\mu^{(u_t^j)}$.

PROOF. Fix any $j \in [k]$ and consider any assignment $(x^{j'})_{j' \in [k] \setminus \{j\}}$ of values $x^{j'} \in \{0,1\}^n$ to the inputs other than the jth input, where $x^{j'}$ extends $u_t^{j'}$ for each $j' \neq j$. We show that after conditioning on the query outcomes $u_t^1, \ldots, u_t^k$ *and* on the event $[\mathbf{x}^{j'} = x^{j'} \forall j' \neq j]$, the jth input $\mathbf{x}^j$ is distributed according to $\mu^{(u_t^j)}$. This will prove the Lemma.

Consider each $y \in \{0,1\}^n$ that extends u_t^j. Now $u_t^1, \ldots, u_t^k$ are, by assumption, a possible description of the first t queries made by $\mathcal{D}$ under *some* input. Since $\mathcal{D}$ is deterministic, and $(x^1, \ldots, x^{j-1}, y, x^{j+1}, \ldots, x^k)$ are consistent with $(u_t^1, \ldots, u_t^k)$, we conclude that $(u_t^1, \ldots, u_t^k)$ also describe the first t queries made by $\mathcal{D}$ on $(x^1, \ldots, x^{j-1}, y, x^{j+1}, \ldots, x^k)$. Thus, the conditional probability that $\mathbf{x}^j = y$ is

$$\begin{aligned}
&\frac{\mu^{\otimes k}(x^1, \ldots, x^{j-1}, y, x^{j+1}, \ldots, x^k)}{\sum_{z \text{ extends } u_t^j} \mu^{\otimes k}(x^1, \ldots, x^{j-1}, z, x^{j+1}, \ldots, x^k)} \\
&\quad = \frac{\mu(y) \cdot \prod_{j' \neq j} \mu(x^{j'})}{\sum_{z \text{ extends } u_t^j} \mu(z) \cdot \prod_{j' \neq j} \mu(x^{j'})} \\
&\quad = \frac{\mu(y)}{\sum_{z \text{ extends } u_t^j} \mu(z)} = \mu^{(u_t^j)}(y),
\end{aligned}$$

by definition of $\mu^{(u_t^j)}$. This proves Lemma 3.2. □

Next, define collections

$$\mathcal{X} = \{X_{j,t}\}_{j \in [k], 0 \leq t \leq M}, \quad \mathcal{P} = \{P_t\}_{0 \leq t \leq M}$$

of random variables, as follows. All the random variables are determined by the execution of $\mathcal{D}$ on an input drawn from $\mu^{\otimes k}$. Let $X_{j,t} := W(u_j^t)$ if $|u_j^t| \leq T$; otherwise let $X_{j,t} := 1/2$. Let $P_t := \prod_{j \in [k]} X_{j,t}$.

We claim that for each $0 \leq t < M$, $\mathbb{E}[P_{t+1}] \leq \mathbb{E}[P_t]$. To see this, condition on any outcomes to the first t queries, described by $u_t^1, \ldots, u_t^k$. Now suppose that for the $(t+1)$st query, $\mathcal{D}$ queries the ith bit of the jth input (i, j are determined by $u_t^1, \ldots, u_t^k$, since $\mathcal{D}$ is deterministic). We note that $X_{j',t+1} = X_{j',t}$ for all $j' \neq j$. If $|u_j^t| \geq T$ then also $X_{j,t+1} \leq X_{j,t}$, which implies $P_{t+1} \leq P_t$. So

assume $|u_j^t| < T$. Then we have

$$\begin{aligned}\mathbb{E}[P_{t+1}|u_t^1,\ldots,u_t^k] &= \mathbb{E}[X_{j,t+1}\cdot\prod_{j'\neq j}X_{j',t+1}|u_t^1,\ldots,u_t^k]\\ &= \mathbb{E}[X_{j,t+1}|u_t^1,\ldots,u_t^k]\cdot\prod_{j'\neq j}X_{j',t}\\ &\leq X_{j,t}\cdot\prod_{j'\neq j}X_{j',t} = P_t,\end{aligned}$$

where we used Lemma 3.2 and part 2 of Lemma 3.1. We conclude

$$\mathbb{E}[P_{t+1}] = \mathbb{E}[\mathbb{E}[P_{t+1}|u_t^1,\ldots,u_t^k]] \leq \mathbb{E}[P_t],$$

as claimed. It follows that $\mathbb{E}[P_M] \leq \mathbb{E}[P_0]$. But we can bound P_0 directly: $P_0 = W(*^n)^k \leq (1-\varepsilon)^k$ (Lemma 3.1, part 1). Thus $\mathbb{E}[P_M] \leq (1-\varepsilon)^k$.

Now we argue that this implies an upper bound on the success probability of $\mathcal{D}$. Condition on the bits $u_M^1,\ldots,u_M^k$ seen by $\mathcal{D}$ during a complete execution; these determine the k output bits of $\mathcal{D}$. For each $j \in [k]$, at least one of two possibilities holds: Either $|u_M^j| > T$, or the jth input is in a final state $\mu^{(u_M^j)}$ for which $\Pr_{\mathbf{y}\sim\mu^{(u_M^j)}}[f(\mathbf{y})=1] \in [1-X_{j,M}, X_{j,M}]$. Since the k inputs remain independent under our conditioning, the conditional probability that $\mathcal{D}$ computes $f^{\otimes k}$ correctly is at most $\prod_{j:|u_M^j|\leq T} X_{j,M}$.

$\mathcal{D}$ makes at most $\alpha\varepsilon Tk$ queries, so simple counting tells us that there are fewer than $\alpha\varepsilon k$ indices j for which $|u_M^j| > T$. Thus,

$$\prod_{j:|u_M^j|\leq T} X_{j,M} \leq \frac{\prod_{j\in[k]} X_j}{(\min_{j\in[k]} X_{j,M})^{\alpha\varepsilon k}} \leq 2^{\alpha\varepsilon k} P_M$$

(since $X_{j,M} \geq 1/2$ for all j). Taking expectations, we find that the *overall* success probability of $\mathcal{D}$ is at most $\mathbb{E}[2^{\alpha\varepsilon k}P_M] \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k$.

Finally, we simplify our bound. We claim $2^x < 1 + .84x$ on $(0, 1/2]$. To see this, just note that $2^0 = 1$, that $2^{1/2} < 1.42 = 1 + .84(1/2)$, and that 2^x is a convex function on $\mathbb{R}$. Then, since $0 < \alpha\varepsilon \leq 1/2$, we have $2^{\alpha\varepsilon}(1-\varepsilon) < (1+.84\alpha\varepsilon)(1-\varepsilon) < 1-\varepsilon+.84\alpha\varepsilon$. The proof is complete. □

We remark that, as claimed in the Introduction, the proof above can be easily adapted to give an alternate proof of Shaltiel's optimal direct product theorem for "fair" algorithms making Tk queries: We define the random variables $X_{j,t}$ exactly as before and note that $|u_t^j| \leq T$ for all j, t.

4. Tightness of the bounds in Theorem 1.1

In this section, we describe a family of functions and input distributions, due to Shaltiel (2003), and explain why they show that the query/success trade-off in Theorem 1.1 is nearly best-possible, at least when $\alpha < .01$ and when $(1-\varepsilon)^k$ is also at most a small constant.

Fixing an integer $T > 0$, define $f_T : \{0,1\}^{T+2} \to \{0,1\}$ as follows: let $f_T(x) := x_2$ if $x_1 = 1$, otherwise $f_T(x) := x_2 \oplus \cdots \oplus x_{T+2}$. Given $\varepsilon \in (0, 1/2)$, let μ_ε be the distribution over $\{0,1\}^{T+2}$ in which all bits are independent, $\Pr[x_1 = 1] = 1 - 2\varepsilon$, and $\Pr[x_i = 1] = 1/2$ for all $i \in \{2, \ldots, T+2\}$. Note that if $\mathbf{y} \sim \mu_\varepsilon$, a T-query-bounded algorithm can gain no information about the value of f when $x_1 = 0$, so any such algorithm succeeds with probability at most $(1-2\varepsilon)1 + (2\varepsilon)\frac{1}{2} = 1 - \varepsilon$ in computing $f(\mathbf{y})$.

Now consider the following algorithm $\mathcal{D}$ attempting to compute $f^{\otimes k}$ on inputs $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu_\varepsilon^{\otimes k}$. First, $\mathcal{D}$ queries the first two bits of each input. Call an input $\mathbf{x}^k$ "bad" if its first bit is 0, and "good" if its first bit is 1. Let $B \subseteq [k]$ denote the set of bad inputs. Note that $\mathcal{D}$ learns the value of f on each good input. Next, $\mathcal{D}$ chooses arbitrarily a set $S \subseteq B$ of $\lfloor \alpha\varepsilon k \rfloor$ bad inputs and spends T additional queries on each input in S to determine the value of f on these inputs (if there are fewer than $\lfloor \alpha\varepsilon k \rfloor$ bad inputs, $\mathcal{D}$ queries them all and determines the value of $f^{\otimes k}$ with certainty). Finally, $\mathcal{D}$ outputs the answer bits it has learned and makes random guesses for the remaining values.

Observe that $\mathcal{D}$ uses at most $2k + \alpha\varepsilon Tk$ queries overall. To analyze the success probability of $\mathcal{D}$, first consider an algorithm $\mathcal{D}'$ which uses only $2k$ queries to look at the two bits of each input; $\mathcal{D}'$ outputs the correct value on good inputs and guesses randomly on bad inputs. It is easy to see that $\mathcal{D}'$ succeeds with probability $(1-\varepsilon)^k$ in computing $f^{\otimes k}$. Also, if both $\mathcal{D}$ and $\mathcal{D}'$ are run on a

common k-tuple of inputs drawn from $\mu_\varepsilon^{\otimes k}$, and we condition on the event that $|B| \geq \lfloor \alpha\varepsilon k \rfloor$, then the success probability of $\mathcal{D}$ is $2^{\lfloor \alpha\varepsilon k \rfloor}$ times the success probability of $\mathcal{D}'$, since the inputs are independent and $\mathcal{D}$ has $\lfloor \alpha\varepsilon k \rfloor$ fewer random guesses to make. Thus, $\Pr[\mathcal{D} \text{ succeeds}]$ is at least

$$\begin{aligned} &\Pr[|B| \geq \alpha\varepsilon k] \cdot 2^{\lfloor \alpha\varepsilon k \rfloor} \Pr\left[\mathcal{D}' \text{ succeeds} \middle| |B| \geq \alpha\varepsilon k\right] \\ &\quad = 2^{\lfloor \alpha\varepsilon k \rfloor} \Pr[\mathcal{D}' \text{ succeeds} \wedge |B| \geq \alpha\varepsilon k] \\ &\quad \geq 2^{\lfloor \alpha\varepsilon k \rfloor} \cdot (\Pr[\mathcal{D}' \text{ succeeds}] - \Pr[|B| < \alpha\varepsilon k]) \\ &\quad = 2^{\lfloor \alpha\varepsilon k \rfloor} \cdot \left((1-\varepsilon)^k - \Pr[|B| < \alpha\varepsilon k]\right). \end{aligned} \tag{4.1}$$

Define the indicator variable $Y_j := \mathbf{1}_{[j \notin B]}$; then the Y_j's are independent, with $p = \Pr[Y_j = 1] = 1 - 2\varepsilon$. Let $Y := Y_1 + \cdots + Y_k$. We apply Lemma 2.2 to Y, with the settings $\delta := 2\varepsilon$ and $\beta := \alpha/2 \leq 1/2$, to obtain

$$\begin{aligned} \Pr[|B| < \alpha\varepsilon k] &= \Pr[Y > (1 - \alpha\varepsilon)k] \\ &= \Pr[Y > (1 - (2\varepsilon)(\alpha/2))k] \\ &< \left[1 - 2\varepsilon + 6(\alpha/2)\ln(2/\alpha)(2\varepsilon)\right]^k. \end{aligned}$$

This can be made less than $(1 - 1.5\varepsilon)^k$ if α is a small enough positive constant ($\alpha < .01$ will work).

Now if $(1-\varepsilon)^k$ is also at most a sufficiently small constant, then $(1 - 1.5\varepsilon)^k < .1(1-\varepsilon)^k$ so that, by Eq. (4.1),

$$\Pr[\mathcal{D} \text{ succeeds}] > .9 \cdot 2^{\lfloor \alpha\varepsilon k \rfloor}(1-\varepsilon)^k,$$

which is close to the maximum success probability allowed by Theorem 1.1 if $\mathcal{D}$ used $\alpha\varepsilon Tk$ queries. (Recall, though, that $\mathcal{D}$ uses $2k + \alpha\varepsilon Tk$ queries.)

5. Proof of Theorem 1.2

We now prove Theorem 1.2 from the Introduction, our DPT for worst-case error, by combining Theorem 1.1 with a version of Yao's minimax principle (Yao 1977), which allows us to convert worst-case hardness assumptions in query complexity into average-case assumptions.

Define $R_{2,\delta}(f)$ as the minimum T for which there exists a randomized T-query algorithm which computes $f(x)$ correctly with probability at least $1-\delta$ for every x. The following is a common version of Yao's principle and can be proved directly using the minimax theorem of game theory.

LEMMA 5.1. Fix $0 < \delta < 1/2$ and a Boolean function f. There exists a distribution μ_δ over inputs to f, such that every randomized algorithm making fewer than $R_{2,\delta}(f)$ queries succeeds in computing f on μ_δ with probability less than $1-\delta$.

Proof of Theorem 1.2 Let f be given. Let $\delta := 1/2 - \gamma/2$, and let $\mu := \mu_\delta$ be as provided by Lemma 5.1. Now fix a tiny constant $c \in (0,1)$, and let $T := R_{2,\delta}(f) - c$; we have

$$\mathrm{Suc}_{T,\mu}(f) \leq 1 - \varepsilon,$$

for some value $\varepsilon > \delta > 3/8$ (independent of c). Now set $\alpha := \gamma$, and apply Theorem 1.1 to find

$$\mathrm{Suc}_{\gamma\varepsilon Tk,\mu}(f) < (1-(1-.84\gamma)\varepsilon)^k < (1-(1-.84\gamma)\delta)^k.$$

Note that $\gamma\varepsilon Tk > \gamma\delta R_{2,\delta}(f)k$, if c is chosen sufficiently small. We conclude that any algorithm making at most $\gamma\delta R_{2,\delta}(f)k$ queries succeeds with probability less than

$$\begin{aligned}(1-(1-.84\gamma)\delta)^k &= (1-(1-.84\gamma)(1/2-\gamma/2))^k \\ &< (1/2+.42\gamma+\gamma/2)^k < (1/2+\gamma)^k\end{aligned}$$

in computing $f^{\otimes k}$ on inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$. So, the worst-case success probability is also less than this amount.

Now we relate $R_{2,\delta}(f)$ to $R_2(f)$ by standard sampling ideas. Say $\mathcal{R}_\delta$ is an algorithm making $R_{2,\delta}(f)$ queries, which computes $f(x)$ with probability at least $1-\delta = 1/2+\gamma/2$ on each input. Let $\mathcal{R}$ be the algorithm that given an input x runs $\mathcal{R}_\delta(x)$ for $m := \lceil 3/\gamma^2 \rceil$ trials, outputting the majority value. For $i \in [m]$, define the indicator variable Y_i for the event [$\mathcal{R}_\delta$ succeeds on the ith trial], and let $Y := Y_1 + \cdots + Y_m$. Then the probability that $\mathcal{R}(x)$ outputs an incorrect value is at most the probability that $Y \leq \mathbb{E}[Y] - \gamma m/2$, which by Hoeffding's inequality is at most $e^{-2\gamma^2 m/4} \leq e^{-3/2} < 1/3$.

Thus, $R_2(f) \leq R_{2,\delta}(f) \cdot \lceil 3/\gamma^2 \rceil < 4R_{2,\delta}(f)/\gamma^2$ (using $\gamma < 1/4$). Then, we have

$$\gamma^3 R_2(f)k/11 < \gamma(3/8)(\gamma^2 R_2(f)/4)k < \gamma\delta R_{2,\delta}(f)k,$$

from which Theorem 1.2 follows.

6. Threshold direct product theorems

In this section, we prove our "gambling lemma," Lemma 6.1, and use it to prove generalized threshold DPTs for relations (relation problems are formally defined in Section 6.2). This will yield DPTs for non-Boolean functions as well as for the query complexity of learning tasks. Further applications of Lemma 6.1 will appear in later sections.

Let $\mathcal{P}([k])$ denote the collection of subsets of $[k]$. Say that a subcollection $\mathcal{A} \subseteq \mathcal{P}([k])$ is *monotone* if $[A \in \mathcal{A},\ A \subseteq A']$ implies $A' \in \mathcal{A}$. Monotone collections play an important role in what follows.

6.1. A gambling lemma. Like the proof of Theorem 1.1, the statement of our next lemma is best explained by a gambling metaphor. Suppose that a gambler gambles at k tables, bringing an initial endowment of $p_j \in [0,1]$ to the jth table. He cannot transfer funds between tables or go into debt at any table; he can only play games for which his expected winnings are nonpositive; and the different tables' games use independent randomness. However, the gambler can choose which game to play next at each table.

The gambler wants to reach a fortune of 1 at "sufficiently many" of the tables, where the meaning of "sufficiently many" is specified by a monotone subset $\mathcal{A} \subseteq \mathcal{P}([k])$. One way the gambler may attempt to reach this goal is to simply place an "all-or-nothing" bet independently at each table; that is, at the jth table, the gambler wins a fortune of 1 with probability p_j and loses his jth endowment with the remaining probability. The following lemma states that this is in fact the gambler's best strategy.

LEMMA 6.1. Suppose $k, N \geq 1$ are given, along with a collection $\{\mathcal{X},\mathcal{U}\}$ of random variables (over a finite probability

space). Here $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$, where for each $j \in [k]$, $\mathcal{X}_j = \{X_{j,0}, X_{j,1}, \ldots, X_{j,N}\}$ is a sequence of variables in the range $[0,1]$ *(think of $X_{j,t}$ as the gambler's fortune at the jth table after the first t steps).* $\mathcal{U} = \{U_0, U_1, \ldots, U_{N-1}\}$ is a sequence of random variables taking values over some finite set *(think of U_t as describing the form and outcomes of all gambles in the first t steps).* Assume that for all $0 \leq t < N$, U_t determines $\{X_{1,t}, \ldots, X_{k,t}\}$, and also determines $U_{t'}$ for all $t' < t$. Also assume that $\{X_{1,t+1}, \ldots, X_{k,t+1}\}$ are independent conditioned on U_t. Then, if $X_{j,0} \leq p_j \in [0,1]$ for all $j \in [k]$, and $\mathcal{A}$ is a monotone subset of $\mathcal{P}([k])$, we have

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{A}] \leq \Pr[D \in \mathcal{A}],$$

where $D \subseteq [k]$ is generated by independently including each $j \in [k]$ in D with probability p_j.

Note that we assume the gambler never attains a fortune greater than 1 at any table; this restriction is easily removed, but it holds naturally in the settings where we will apply the Lemma.

Proof. We use the term "$\mathcal{A}$-success" to refer to the event $[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{A}]$ whose probability we are bounding.

We first make a simplifying observation: We claim that it is without loss of generality to assume that between each consecutive times $(t, t+1)$, at most one of the fortunes changes and that the fortune subject to change is determined by t. Call a family of sequences with this property "nice." To see this, consider any family $\mathcal{X}$ obeying Lemma 6.1's assumptions, and modify it by "splitting" each transition $(t, t+1)$ into a sequence of k transitions, in the jth of which the jth fortune changes (according to the same distribution governing its transition in the original sequence).

More formally, we define $\mathcal{X}'_j = \{X'_{j,0}, \ldots, X'_{j,Nk}\}$ by letting $X'_{j,\ell} := X_{j,\lfloor(\ell+k-j)/k\rfloor}$, and we define $\mathcal{U}' = \{U'_0, U'_1, \ldots, U'_{Nk-1}\}$ by

$$U'_\ell := \left(U_{\lfloor \ell/k \rfloor}, \left(X'_{j,\ell'}\right)_{j \in [k], \ell' \leq \ell}\right).$$

(We add extra information into U'_ℓ to ensure that it determines the random variables it is supposed to.) Lemma 6.1's assumptions continue to hold for this modified, nice family of random variables;

here we are using our original assumption that $\{X_{1,t+1}, \ldots, X_{k,t+1}\}$ are independent conditioned on U_t. Also, the probability of $\mathcal{A}$-success is unchanged. So let us assume from now on that $(\mathcal{X},\mathcal{U})$ is nice, and for $0 \leq t < N$, let $j_t \in [k]$ be the index of the fortune subject to change between times t and $t+1$.

Fix any $k \geq 1$; we prove the statement by induction on $N \geq 1$. First suppose $N = 1$, and let j_0 be as defined above. Let $S \subseteq [k] \setminus \{j_0\}$ be the set of indices $j \neq j_0$ for which $p_j = 1$. First suppose $S \in \mathcal{A}$; then $\Pr[D \in \mathcal{A}] = 1$, since each $j \in S$ is included in D with probability 1. In this case, the conclusion is trivially satisfied. Next suppose $S \cup \{j_0\} \notin \mathcal{A}$. In this case, $\Pr[\mathcal{A}\text{-success}] = 0$, and again the conclusion is trivially satisfied. So suppose $S \notin \mathcal{A}$, $S \cup \{j_0\} \in \mathcal{A}$, and condition on any value $U_0 = u$. Then $\mathcal{A}$-success occurs if $X_{j_0,1} = 1$. By Markov's inequality, $\Pr[X_{j_0,1} = 1|U_0 = u] \leq \mathbb{E}[X_{j_0,1}|U_0 = u] \leq X_{j_0,0} \leq p_{j_0} = \Pr[D \in \mathcal{A}]$. This proves the statement for $N = 1$.

So let $N > 1$ and assume the statement proved for $\{1, \ldots, N-1\}$; we prove it for N. Condition on any value $U_0 = u$, and condition further on the value $X_{j_0,1} = a \in [0,1]$. The equalities $X_{j,1} = X_{j,0} \leq p_j$ are forced for all $j \neq j_0$; the residual collection of random variables $\{X_{j,t} : j \in [k], 1 \leq t \leq N\} \cup \{U_t : 1 \leq t < N\}$ under our conditioning obeys Lemma 6.1's assumptions, along with our added assumption; and these sequences are shorter by a step than our initial sequences. Thus, our induction hypothesis implies that

$$\Pr[\mathcal{A}\text{-success}|U_0 = u, X_{j_0,1} = a] \leq \Pr[D^{(a)} \in \mathcal{A}], \tag{6.2}$$

where $D^{(a)}$ is generated just like D except that j_0 is now included in $D^{(a)}$ with probability a.

Let $q_0 := \Pr[D \setminus \{j_0\} \in \mathcal{A}]$ and $q_1 := \Pr[D \cup \{j_0\} \in \mathcal{A}]$. Note that $q_0 \leq q_1$, since $\mathcal{A}$ is monotone. We have

$$\Pr[D^{(a)} \in \mathcal{A}] = (1-a)q_0 + aq_1.$$

Taking expectations over a in Eq. (6.2), $\Pr[\mathcal{A}\text{-success}|U_0 = u]$ is at most

$$\begin{aligned}
&(1 - \mathbb{E}[X_{j_0,1}|U_0 = u])q_0 + \mathbb{E}[X_{j_0,1}|U_0 = u] \cdot q_1 \\
&\quad \leq (1 - p_{j_0})q_0 + p_{j_0}q_1
\end{aligned}$$

$$\begin{aligned}&(\text{since } q_0 \le q_1 \text{ and } \mathbb{E}[X_{j_0,1}|U_0 = u] \le X_{j_0,0} \le p_{j_0})\\ &= \Pr[D \in \mathcal{A}].\end{aligned}$$

As u was arbitrary, this extends the induction to N and completes the proof. □

6.2. Application to threshold DPTs. Now we prove our generalized threshold direct product theorem. Our theorem will be within the framework of solving relation problems, a more general task than computing functions. A *relation* (with Boolean domain) is a subset $P \subseteq \{0,1\}^n \times B$, for some finite set B. The relation is *total* if for all $x \in \{0,1\}^n$, there exists $b \in B$ such that $(x,b) \in P$. For each total relation P, there is a natural computational problem: Given an input x, try to output a b for which $(x,b) \in P$. Computing a function $f : \{0,1\}^n \to B$ is equivalent to solving the relation problem for the total relation $P_f := \{(x,b) : f(x) = b\}$.

If $\mathcal{R}$ is a (possibly randomized) query algorithm producing outputs in B, P is a total relation, and μ a distribution, say that $\mathcal{R}$ *ε-solves P with respect to μ* if $\Pr_{x\sim\mu}[(x,\mathcal{R}(x)) \in P] \ge 1-\varepsilon$. Define $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) := 1-\varepsilon$, where $\varepsilon \ge 0$ is the minimum value for which some T-query randomized algorithm $\mathcal{R}\varepsilon$-solves P with respect to μ. As usual, this minimum exists and is attained by a deterministic height-T decision tree. For a randomized algorithm $\mathcal{R}$ making queries to $k \ge 1$ inputs $x = (x^1,\dots,x^k)$ to P and producing an output in B^k, let $\mathcal{R}_j(x) \in B$ be the jth value outputted by $\mathcal{R}$.

Given $A, A' \subseteq [k]$, define the *distance* $d(A,A') := |(A \setminus A') \cup (A' \setminus A)|$. Given a set family $\mathcal{A} \subseteq \mathcal{P}([k])$, and a real number $r > 0$, define the *strict r-neighborhood of $\mathcal{A}$*, denoted $N_r(\mathcal{A})$, as

$$N_r(\mathcal{A}) := \{A' : d(A,A') < r \text{ for some } A \in \mathcal{A}\}.$$

We have $\mathcal{A} \subseteq N_r(\mathcal{A})$. Note also that if $\mathcal{A}$ is monotone then so is $N_r(\mathcal{A})$. We can now state our generalized threshold DPT:

THEOREM 6.3. Fix a finite set B, and let $P \subseteq \{0,1\}^n \times B$ be a total relation for which $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) \le 1-\varepsilon$. *Fixing any randomized algorithm $\mathcal{R}$ making queries to inputs $\mathbf{x} = (\mathbf{x}^1,\dots,\mathbf{x}^k) \sim \mu^{\otimes k}$ and producing output in B^k, define the (random) set*

$$S[\mathbf{x}] := \{j \in [k] : (\mathbf{x}^j, \mathcal{R}_j(\mathbf{x})) \in P\}.$$

Suppose $\mathcal{R}$ is $\alpha\varepsilon Tk$-query-bounded for some $\alpha \in (0,1]$, and $\mathcal{A}$ is any monotone subset of $\mathcal{P}([k])$. Then:

1. $\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq |B|^{\alpha\varepsilon k} \cdot \Pr[D \in \mathcal{A}]$, *where $D \subseteq [k]$ is generated by independently including each $j \in [k]$ in D with probability $1-\varepsilon$.*

2. *Also, for D as above,* $\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]$.

Proof. As in Theorem 1.1, we may assume $\varepsilon, T > 0$, $\mathrm{supp}(\mu) = \{0,1\}^n$. We have $\varepsilon \leq 1 - |B|^{-1} < 1$, since P is total and an algorithm may output a random element of B.

For $u \in \{0,1,*\}^n$ with $|u| \leq T$, let

$$W_P(u) := \mathrm{Suc}^{\mathrm{rel}}_{T-|u|,\mu^{(u)}}(P).$$

Then $W_P(u) \in [|B|^{-1}, 1]$. We have the following claim, whose proof follows that of Lemma 3.1:

Lemma 6.4. 1. $W_P(*^n) \leq 1 - \varepsilon$.

2. For any $u \in \{0,1,*\}^n$ with $|u| < T$, and any $i \in [n]$, $\mathbb{E}_{\mathbf{y}\sim\mu^{(u)}}[W_P(u[x_i \leftarrow \mathbf{y}_i])] \leq W_P(u)$.

Let $\mathcal{R}$ be $\alpha\varepsilon Tk$-query-bounded; as in Theorem 1.1, we may assume $\mathcal{R}$ is deterministic, so call it $\mathcal{D}$ instead. Let $M := \lfloor \alpha\varepsilon Tk \rfloor$ as before, and recall the random strings u_t^j defined in Theorem 1.1.

Define random variables $\{X_{j,t}\}_{j\in[k],0\leq t\leq M}$, determined by an execution of $\mathcal{D}$ on inputs $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$, by letting $X_{j,t} := W_P(u_t^j)$ if $|u_t^j| \leq T$, otherwise $X_{j,t} := |B|^{-1}$. Next, the natural idea is to apply Lemma 6.1. First, however, we need to extend the sequences for one additional (nonquery) step. That is, we will define random variables $X_{j,M+1}$ for each $j \in [k]$. We will use $\mathcal{X}$ to denote the collection of enlarged sequences.

Our definition of $X_{j,M+1}$ depends on whether $|u_M^j| \leq T$, that is, on whether $\mathcal{D}$ made at most T queries to $\mathbf{x}^j$ on the current execution. If $|u_M^j| \leq T$, let $X_{j,M+1} := \mathbf{1}_{[(\mathbf{x}^j, \mathcal{D}_j(\mathbf{x}))\in P]}$ be the indicator variable for the event that $\mathcal{D}$ solves P on the jth input. If $|u_M^j| > T$, let $X_{j,M+1} := 1$ with probability $|B|^{-1}$, and let $X_{j,M+1} := 0$ with

the remaining probability. We let each such "coin-flip" be independent of the others and of $(\mathbf{x}^1, \ldots, \mathbf{x}^k)$.

Define the collection $\mathcal{U} = \{U_0, \ldots, U_M\}$ by $U_t := (u_t^1, \ldots, u_t^k)$. We argue that the conditions of Lemma 6.1 are satisfied by $(\mathcal{X}, \mathcal{U})$, with $N := M + 1$. First, for $0 \leq t' \leq t \leq M$, the stated conditions follow from Lemma 3.2 and part 2 of Lemma 6.4. Now consider the final, added step. Condition on any value of $U_M = (u_M^1, \ldots, u_M^k)$. Lemma 3.2 tells us that $\mathbf{x}^1, \ldots, \mathbf{x}^k$ are independent under this conditioning, and $\mathcal{D}$'s outputs are determined by U_M, so the variables $\{X_{j,M+1}\}$ are independent conditioned on U_M. If $|u_M^j| \leq T$ then $\mathbb{E}[X_{j,M+1}|U_M] \leq X_{j,M}$ by part 2 of Lemma 6.4. If $|u_M^j| > T$ then $\mathbb{E}[X_{j,M+1}] = |B|^{-1} = X_{j,M}$.

Thus, the assumptions of Lemma 6.1 are satisfied, with $p_j = X_{j,0} \leq 1 - \varepsilon$. We conclude that for any monotone $\mathcal{C} \subseteq \mathcal{P}([k])$,

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{C}] \leq \Pr[D \in \mathcal{C}], \tag{6.5}$$

where each $j \in [k]$ is independently included in D with probability $1 - \varepsilon$.

To prove statement 1 of Theorem 6.3, let $\mathcal{C} := \mathcal{A}$. Note that $S[\mathbf{x}]$ and $u_M^1, \ldots, u_M^k$ are determined by $\mathbf{x}$, since $\mathcal{D}$ is deterministic. Condition on any value of $\mathbf{x}$ for which $S[\mathbf{x}] \in \mathcal{A}$. Under this conditioning, if $j \in [k]$ satisfies $|u_M^j| \leq T$ and $j \in S[\mathbf{x}]$, then $X_{j,N} = 1$. On the other hand, if $|u_M^j| > T$, then $[X_{j,N} = 1]$ holds with probability $|B|^{-1}$, and these events are independent for each such j. By the query bound on $\mathcal{D}$, there are fewer than $\alpha\varepsilon k$ indices j in our conditioning for which $|u_M^j| > T$. Thus,

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{A} | S[\mathbf{x}] \in \mathcal{A}] \geq |B|^{-\alpha\varepsilon k},$$

which in combination with Eq. (6.5) implies

$$\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq |B|^{\alpha\varepsilon k} \cdot \Pr[D \in \mathcal{A}],$$

as needed. To prove statement 2 of Theorem 6.3, let $\mathcal{C} := N_{\alpha\varepsilon k}(\mathcal{A})$ in Eq. (6.5): we find

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})].$$

Arguing as above, $S[\mathbf{x}] \setminus \{j \in [k] : X_{j,N} = 1\}$ is always a set of size less than $\alpha\varepsilon k$, so $[S[\mathbf{x}] \in \mathcal{A}]$ implies $[\{j \in [k] : X_{j,N} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})]$. Thus, we have $\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]$. □

Part 1 of Theorem 6.3 is a proper generalization of Theorem 1.1. To see this, just set $\mathcal{A} := \{[k]\}$, $P := P_f$, and note that in this case, $\Pr[D \in \mathcal{A}] = (1-\varepsilon)^k$. As another dividend, we obtain the following threshold DPT for relations, which specializes to an ordinary DPT for this setting (statement 3 in the Theorem below).

THEOREM 6.6. Let $P \subseteq \{0,1\}^n \times B$ be a total relation for which $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) \leq 1-\varepsilon$. *Fix any* $\eta \in (0,1]$. *For any randomized algorithm* $\mathcal{R}$ *making queries to inputs* $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$, *define the (random) set* $S[\mathbf{x}]$ *as in* Theorem 6.3. *Then if* $\mathcal{R}$ *is* $\alpha\varepsilon Tk$*-query-bounded for* $\alpha \in (0,1]$, *we have:*

1. $\Pr[|S[\mathbf{x}]| \geq \eta k] \leq |B|^{\alpha\varepsilon k} \cdot \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq \eta k]$, *and also*
2. $\Pr[|S[\mathbf{x}]| \geq \eta k] \leq \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (\eta - \alpha\varepsilon)k]$.
3. $\Pr[|S[\mathbf{x}]| = [k]]$ *is at most the minimum of* $|B|^{\alpha\varepsilon k}(1-\varepsilon)^k$ *and* $\Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (1-\alpha\varepsilon)k]$. *If* $\alpha \leq 1/2$ *the second bound in the* min *is at most* $[1 - \varepsilon + 6\alpha \ln(1/\alpha)\varepsilon]^k$.

PROOF. Apply parts 1 and 2 of Theorem 6.3, with the choice $\mathcal{A} := \{A \subseteq [k] : |A| \geq \eta k\}$. We have $\Pr[D \in \mathcal{A}] = \Pr[D_1 + \cdots + D_k \geq \eta k]$, where we define $D_j := \mathbf{1}_{[j \in D]}$. These 0/1-valued variables are independent with bias $1-\varepsilon$, which gives statement 1. Similarly, $\Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})] = \Pr[D_1 + \cdots + D_k \geq (\eta - \alpha\varepsilon)k]$, which gives statement 2. Statement 3 simply combines statements 1 and 2, under the setting $\eta = 1$. For the final bound in statement 3, we apply Lemma 2.2 with $\beta := \alpha$, $\delta := \varepsilon$. □

Theorem 1.4 in the Introduction follows from the special case of Theorem 6.6 in which $P := P_f$.

The success bound $|B|^{\alpha\varepsilon k}(1-\varepsilon)^k$ appearing above can also be derived by an easy modification of the proof of Theorem 1.1, in which the condition $X_{j,t} \geq 1/2$ we exploit becomes $X_{j,t} \geq |B|^{-1}$. When $|B|$ is large, however, the alternative bound provided in Theorem 6.6 will tend to give better results.

Note that part 2 of Theorem 6.6, in conjunction with Chernoff inequalities, gives success bounds which decay exponentially in k for any fixed $\alpha, \varepsilon, \eta$ for which $\eta > 1 - \varepsilon + \alpha\varepsilon$. Shaltiel's examples, described in Section 4, show that this cutoff is nearly tight:

On those functions, the algorithm $\mathcal{D}$ described in Section 4 makes $2k + \alpha\varepsilon Tk$ queries and (it is easily checked) typically solves about $(1 - \varepsilon + .5\alpha\varepsilon)k$ of the instances correctly.

Threshold DPTs for the worst-case setting can also be derived from Theorems 6.3 and 6.6, by the same reduction to the average-case setting used to prove Theorem 1.2.

6.3. Direct product theorems for learning tasks. Theorems 6.3 and 6.6 readily imply direct product theorems for the query complexity of certain learning tasks, as we explain next. Consider the scenario in which a randomized algorithm $\mathcal{R}$ is given query access to an unknown function $h : \{0,1\}^n \to \{0,1\}$ drawn from some distribution μ over a hypothesis class $\mathcal{H}$. That is, for any string x, $\mathcal{R}$ can query the value $h(x)$. The algorithm $\mathcal{R}$ attempts to output a hypothesis $\tilde{h}$ which is "close" to h. That is, we fix some symmetric relation $\mathbf{close} \subseteq \mathcal{H} \times \mathcal{H}$ (assume $\mathbf{close}(h,h)$ always holds), and we wish to find some $\tilde{h}$ such that $\mathbf{close}(h,\tilde{h})$ holds.

This task can be equivalently modeled as the relation problem associated with the total relation

$$P_{\mathcal{H}} := \{(h,h') : h,h' \in \mathcal{H} \wedge \mathbf{close}(h,h')\},$$

where h is given in truth-table form as a Boolean string, under the input distribution $h \sim \mu$. (We do not give a membership criterion for $P_{\mathcal{H}}$ when $h \notin \mathcal{H}$; this is unimportant since $\mathrm{supp}(\mu) \subseteq \mathcal{H}$.)

In the k-fold learning problem associated with $\mathcal{H}, \mu$, the algorithm has query access to each of k functions $(h_1, \ldots, h_k) \sim \mu^{\otimes k}$, and the goal is to output guesses $\tilde{h}_1, \ldots, \tilde{h}_k$ such that $\mathbf{close}(h_j, \tilde{h}_j)$ holds for all (or at least "many") indices $j \in [k]$. This task is equivalent to the k-fold relation problem associated with $P_{\mathcal{H}}$, and Theorems 6.3 and 6.6 apply.

7. Proof of the XOR lemma

The proof of our XOR Lemma, Theorem 1.3 from the Introduction, is modeled on the proof of our threshold DPTs and reuses Lemma 6.1.

Proof of Theorem 1.3. As usual we first set up some preliminaries. For a deterministic algorithm $\mathcal{D}$ over n input bits, define

$$W_{\oplus}(u) := 2 \cdot \mathrm{Suc}_{T-|u|,\mu^{(u)}}(f) - 1.$$

LEMMA 7.1. 1. $W_{\oplus}(*^n) \leq 1 - 2\varepsilon$.

2. For any $u \in \{0,1,*\}^n$ with $|u| < T$, and any $i \in [n]$, $\mathbb{E}_{\mathbf{y}\sim\mu^{(u)}}[W_{\oplus}(u[x_i \leftarrow \mathbf{y}_i])] \leq W_{\oplus}(u)$.

Lemma 7.1 follows immediately from Lemma 3.1, since $W_{\oplus}(u) = 2W(u) - 1$.

Now we prove the Theorem. As in the proof of Theorem 1.1, we may assume $\varepsilon, T > 0$, $\mathrm{supp}(\mu) = \{0,1\}^n$, and it is enough to prove the success bound for each deterministic $\alpha\varepsilon Tk$-query algorithm $\mathcal{D}$ attempting to solve $f^{\oplus k}(\mathbf{x}^1, \ldots, \mathbf{x}^k)$ on inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$. Recall the definitions of u_t^j (for $j \in [k], 0 \leq t \leq M := \lfloor \alpha\varepsilon Tk \rfloor$) from Theorem 1.1. For a deterministic algorithm $\mathcal{D}$, define $\{X_{j,t}\}_{j\in[k],0\leq t\leq M}$ as follows: if $|u_t^j| \leq T$, set $X_{j,t} := W_{\oplus}(u_j^t)$; otherwise, set $X_{j,t} := 0$.

We will extend the random sequences $\{X_{j,t}\}$ for one additional (nonquery) step and will let $\mathcal{X}$ denote our enlarged collection. To set up our extension, we first define random variables b_j, r_j, a_j for $j \in [k]$, determined by u_M^j, as follows. Let $b_j \in \{0,1\}$ be defined as the likeliest value of $f(\mathbf{y})$, where $\mathbf{y} \sim \mu^{(u_M^j)}$ (break ties arbitrarily). Let $r_j := \Pr[f(\mathbf{y}) = b_j] \in [1/2, 1]$, where again $\mathbf{y} \sim \mu^{(u_M^j)}$. Let $a_j := 2r_j - 1 \in [0,1]$.

If $|u_M^j| > T$, set $X_{j,M+1} := 0$. If instead $|u_M^j| \leq T$, our random process "inspects" the actual value of the bit $f(\mathbf{x}^j)$ to help determine $X_{j,M+1}$. If $f(\mathbf{x}^j) \neq b_j$, let $X_{j,M+1} := 0$. If $f(\mathbf{x}^j) = b_j$, let $X_{j,M+1} := 1$ with probability a_j/r_j, and $X_{j,M+1} := 0$ with the remaining probability, where this random decision is independent of all others. Thus in this case,

$$\mathbb{E}[X_{j,M+1} | u_M^1, \ldots, u_M^k] = r_j \cdot (a_j/r_j) = a_j \leq X_{j,M},$$

where the last inequality holds by the definition of $W_{\oplus}(u_M^j)$ since $|u_M^j| \leq T$.

Let $\mathcal{U} = (U_0, \ldots, U_M)$, where $U_t := (u_t^1, \ldots, u_t^k)$. By an argument analogous to that in the proof of Theorem 6.3, we verify that $(\mathcal{X}, \mathcal{U})$ obey the assumptions of Lemma 6.1, this time with $p_j := 1 - 2\varepsilon$. Applying Lemma 6.1 to $\mathcal{A} := \{A \subseteq [k] : |A| > (1-\alpha\varepsilon)k\}$, we find

$$\Pr[|\{j : X_{j,M+1} = 1\}| > (1 - \alpha\varepsilon)k] \leq \Pr[D \in \mathcal{A}], \tag{7.2}$$

where each $j \in [k]$ is independently included in D with probability $(1 - 2\varepsilon)$. We have $\Pr[D \in \mathcal{A}] = \Pr_{Y \sim B_{k,1-2\varepsilon}}[Y > (1 - \alpha\varepsilon)k]$.

We analyze events F of form $F := [U_M = (u_M^1, \ldots, u_M^k), X_{1,M+1} = z_1, \ldots, X_{k,M+1} = z_k]$. Note that conditioning on F does *not* condition on the particular values $f(\mathbf{x}^j)$ that helped determine the values z_j. Focus attention on any such event F for which $|\{j : X_{j,M+1} = 1\}| \leq (1 - \alpha\varepsilon)k$. Since $\mathcal{D}$ makes at most $\alpha\varepsilon Tk$ queries, there are fewer than $\alpha\varepsilon k$ indices j for which $|u_M^j| > T$. In particular, there exists a $j^\star \in [k]$ for which $|u_M^{j^\star}| \leq T$ *and* $X_{j^\star,M+1} < 1$ (so, by our definitions, $X_{j^\star,M+1} = 0$).

Now let the event F' be defined just like F, except that F' makes no conditioning on $X_{j^\star,M+1}$ (so, $F = F' \wedge [X_{j^\star,M+1} = 0]$). Then,

$$\begin{aligned}
\Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star} | F] &= \Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star} | F' \wedge X_{j^\star,M+1} = 0] \\
&= \frac{\Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star} \wedge X_{j^\star,M+1} = 0 | F']}{\Pr[X_{j^\star,M+1} = 0 | F']} \\
&= \frac{\Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star} | F'] \cdot \Pr[X_{j^\star,M+1} = 0 | F', f(\mathbf{x}^{j^\star}) = b_{j^\star}]}{\sum_{b \in \{0,1\}} \Pr[f(\mathbf{x}^{j^\star}) = b | F'] \cdot \Pr[X_{j^\star,M+1} = 0 | F', f(\mathbf{x}^{j^\star}) = b]} \\
&= \frac{r_{j^\star}(1 - a_{j^\star}/r_{j^\star})}{r_{j^\star}(1 - a_{j^\star}/r_{j^\star}) + (1 - r_{j^\star}) \cdot 1}
\end{aligned}$$

(using the fact that $\mathbf{x}^1, \ldots, \mathbf{x}^k$ are independent conditioned on U_M, by Lemma 3.2, and the additional fact that $\{X_{j,M+1}\}_{j \in [k]}$ are independent conditioned on U_M)

$$= \frac{r_{j^\star} - a_{j^\star}}{1 - a_{j^\star}} = \frac{\frac{1}{2}(1 + a_{j^\star}) - a_{j^\star}}{1 - a_{j^\star}} = 1/2.$$

Thus, $f(\mathbf{x}^{j^\star})$ is an unbiased random bit conditioned on F. Consequently, $f^{\oplus k}(\mathbf{x}^1, \ldots, \mathbf{x}^k) = f(\mathbf{x}^{j^\star}) \oplus f^{\oplus k-1}(\mathbf{x}^1, \ldots, \mathbf{x}^{j^\star-1}, \mathbf{x}^{j^\star+1},$

$\ldots, \mathbf{x}^k)$ is an unbiased random bit conditioned on F. Thus, under this conditioning, $\mathcal{D}$'s output bit equals the k-fold XOR with probability exactly $1/2$. Now F was an arbitrary outcome of $U_M, X_{1,M+1}, \ldots, X_{k,M+1}$ for which $|\{j : X_{j,M+1} = 1\}| \leq (1-\alpha\varepsilon)k$. It follows that

$$\begin{aligned}
\Pr_{\mathbf{x}\sim\mu^{\otimes k}}[\mathcal{D}(\mathbf{x}) = f^{\oplus k}(\mathbf{x})] &\leq \Pr\left[|\{j : X_{j,M+1} = 1\}| > (1-\alpha\varepsilon)k\right] \\
&\quad + \frac{1}{2}\Pr\left[|\{j : X_{j,M+1} = 1\}| \leq (1-\alpha\varepsilon)k\right] \\
&= \frac{1}{2}\left(1 + \Pr\left[|\{j : X_{j,M+1} = 1\}| > (1-\alpha\varepsilon)k\right]\right) \\
&\leq \frac{1}{2}\left(1 + \Pr_{Y\sim B_{k,1-2\varepsilon}}[Y > (1-\alpha\varepsilon)k]\right),
\end{aligned}$$

using Eq. (7.2).

Finally, to get the concrete bound claimed in statement of Theorem 1.3, first suppose $\varepsilon = 1/2$; in this case, the bound follows easily since $Y = 0$ with certainty. If $\varepsilon < 1/2$, note that $(1-\alpha\varepsilon)k = (1-(\alpha/2)(2\varepsilon))$, and apply Lemma 2.2 with $\delta := 2\varepsilon < 1$ and $\beta := \alpha/2 \leq 1/2$.

8. Direct product theorems for search problems and errorless heuristics

We define a fairly general notion of search problems in the query model for which a direct product theorem can be proved. We will also obtain a DPT for *errorless heuristics*, defined in Section 8.2.

8.1. Search problems. We need some preliminary definitions. Given $u, v \in \{0,1,*\}^n$, say that u and v *agree* if $u_i \in \{0,1\}$ implies $v_i \in \{*, u_i\}$. Note that this definition is symmetric in u and v. If u, v agree, define their *overlay* $u \circ v \in \{0,1,*\}^n$ by $(u \circ v)_i := b \in \{0,1\}$ if either $u_i = b$ or $v_i = b$, otherwise $(u \circ v)_i := *$. Say that u *extends* v if $v_i \in \{0,1\}$ implies $u_i = v_i$.

Say we are given a distribution μ on $\{0,1\}^n$, and a (possibly randomized) query algorithm $\mathcal{R}$; if $\mathcal{R}$ runs on an input distributed according to μ, we denote by $U_{\mathcal{R},\mu} \in \{0,1,*\}^n$ the random string describing the input bits seen by $\mathcal{R}$.

A *search problem* is defined by a subset $V \subseteq \{0,1,*\}^n$. We say that $\mathcal{R}$ *ε-solves* the search problem V with respect to an input distribution μ over $\{0,1\}^n$ if, with probability $\geq 1-\varepsilon$, $U_{\mathcal{R},\mu}$ extends some $v \in V$. (We allow the possibility that some $x \in \mathrm{supp}(\mu)$ do not extend *any* $v \in V$.) Define $\mathrm{Suc}_{T,\mu}(V) := 1-\varepsilon$, where ε is the minimal value such that some T-query randomized algorithm ε-solves search problem V on inputs from μ.

Define the k-fold search problem $V^{\otimes k} := \{(v^1,\ldots,v^k) : v^j \in V, \forall j \in [k]\} \subseteq \{0,1,*\}^{kn}$. Thus to solve $V^{\otimes k}$, an algorithm must solve each of the k constituent search problems. We generalize this notion in order to state a threshold DPT, which will imply our ordinary DPT. For a monotone subset $\mathcal{A} \subseteq \mathcal{P}([k])$, define

$$V^{k,\mathcal{A}} := \{(v^1,\ldots,v^k) : \{j \in [k] : v^j \in V\} \in \mathcal{A}\}.$$

Thus to solve $V^{k,\mathcal{A}}$, an algorithm must solve "sufficiently many" of the k search problems, as specified by $\mathcal{A}$.

Recall the notation $N_r(\cdot)$ from Section 6. Our generalized threshold DPT for search problems is as follows:

THEOREM 8.1. Suppose the search problem V satisfies $\mathrm{Suc}_{T,\mu}(V) \leq 1-\varepsilon$. Then for any $\alpha \in (0,1]$ and any monotone $\mathcal{A} \subseteq \mathcal{P}([k])$,

$$\mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(V^{k,\mathcal{A}}) \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})],$$

where each $j \in [k]$ is independently included in D with probability $1-\varepsilon$.

PROOF. In the search setting, ε can potentially be any value in $[0,1]$. The boundary cases are trivial, so assume $0 < \varepsilon < 1$. As usual, we can assume that $T > 0$ and $\mathrm{supp}(\mu) = \{0,1\}^n$, and it is enough to bound the success probability of any deterministic $\alpha\varepsilon Tk$-query algorithm.

Following Theorem 1.1, we first develop some concepts related to a computation on a single input to the search problem V. For each $u \in \{0,1,*\}^n$ for which $|u| \leq T$, let $\mathrm{Val}_V(u) := 1$ if u extends some $v \in V$, otherwise $\mathrm{Val}_V(u) := 0$. For a deterministic query algorithm $\mathcal{D}$, let $W_V(u,\mathcal{D}) := \mathbb{E}[\mathrm{Val}(u \circ U_{\mathcal{D},\mu^{(u)}})]$. (Note that u and $U_{\mathcal{D},\mu^{(u)}}$ always agree.)

If $|u| \leq T$, let $W_V(u) := \max_{\mathcal{D}}(W_V(u, \mathcal{D}))$, where the maximum ranges over all deterministic algorithms making at most $T - |u|$ queries. In other words, $W_V(u)$ is the maximum success probability of any $(T - |u|)$-query algorithm in solving V on an input $\mathbf{y} \sim \mu^{(u)}$, where we reveal the bits described by u "for free" to the algorithm. Then we have:

LEMMA 8.2. 1. $W_V(*^n) \leq 1 - \varepsilon$.

1. For any $u \in \{0, 1, *\}^n$ with $|u| < T$, and any $i \in [n]$, $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W_V(u[x_i \leftarrow \mathbf{y}_i])] \leq W_V(u)$.

We omit the proof that is essentially the same as that of Lemma 3.1.

Let $\mathcal{D}$ be any deterministic algorithm making at most $M := \lfloor \alpha\varepsilon Tk \rfloor$ queries and attempting to compute $V^{k,\mathcal{A}}$ on inputs drawn as $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$. For $0 \leq t \leq M$, and for $j \in [k]$, let u_t^j be defined as in the previous proofs. Let $\mathcal{X} = \{X_{j,t}\}_{j \in [k], 0 \leq t \leq M}$, where $X_{j,t} := W_V(u_t^j)$ if $|u_t^j| \leq T$, otherwise $X_{j,t} := 0$.

Unlike in Theorem 6.3, we have no need to add any additional steps to our random sequences. For $0 \leq t < M$, we let $U_t := (u_t^1, \ldots, u_t^k)$ just as before. Setting $N := M$ and reasoning as in Theorem 6.3, we verify that the assumptions of Lemma 6.1 are satisfied, with $p_j = X_{j,0} \leq 1 - \varepsilon$ (Lemma 8.2, part 1).

Applying Lemma 6.1 to the monotone set $N_{\alpha\varepsilon k}(\mathcal{A})$, we conclude that

$$\Pr[\{j \in [k] : X_{j,M} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})], \tag{8.3}$$

where each $j \in [k]$ is independently included in D with probability $1 - \varepsilon$.

Now condition on any execution of $\mathcal{D}$, and consider any $j \in [k]$ such that $X_{j,M} < 1$. By our definitions, at least one of two possibilities holds: Either $|u_M^j| > T$ (there are fewer than $\alpha\varepsilon k$ such indices j) or u_M^j does not extend any $v \in V$. Thus, if $\mathcal{D}$ solves the search problem $V^{k,\mathcal{A}}$ on the present execution, we have $\{j \in [k] : X_{j,M} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})$. Combining this with Eq. (8.3) yields the Theorem. □

From Theorem 8.1, we will directly obtain a standard threshold DPT and an ordinary DPT for search problems. First, given a search problem $V \subseteq \{0,1,*\}^n$ and a real number $s \in [0,k]$, define $\mathcal{C}[\geq s] := \{A \subseteq [k] : |A| \geq s\}$.

THEOREM 8.4. Suppose $\mathrm{Suc}_{T,\mu}(V) \leq 1-\varepsilon$. Then for any $\alpha \in (0,1]$ and any $\eta \in (0,1]$,

$$\mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(V^{k,\mathcal{C}[\geq \eta k]}) \leq \Pr_{Y\sim B_{k,1-\varepsilon}}[Y > (\eta-\alpha\varepsilon)k]\,.$$

PROOF. Apply Theorem 8.1 with $\mathcal{C} := \mathcal{C}[\geq \eta k]$, and note that $D \in N_{\alpha\varepsilon k}(\mathcal{C}[\geq \eta k])$ iff $|D| > \eta k - \alpha\varepsilon k$, which is equivalent to $[D_1 + \cdots + D_k > (\eta-\alpha\varepsilon)k]$, where $D_j := \mathbf{1}_{[j\in D]}$. These indicator variables are independent with expectation $1-\varepsilon$. □

THEOREM 8.5. Suppose $\mathrm{Suc}_{T,\mu}(V) \leq 1-\varepsilon$. Then for any $\alpha \in (0,1]$,

$$\mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(V^{\otimes k}) \leq \Pr_{Y\sim B_{k,1-\varepsilon}}[Y > (1-\alpha\varepsilon)k]\,.$$

PROOF. Note that $V^{\otimes k} = V^{k,\mathcal{C}[\geq k]}$, so the result follows from Theorem 8.4 with $\eta := 1$. □

8.2. Errorless heuristics. An *errorless heuristic* for a (not necessarily Boolean) function f is a randomized query algorithm $\mathcal{R}$ outputting values in $\{0,1,?\}$ such that for all x, $\mathcal{R}(x) \in \{f(x),?\}$ with probability 1. We say that an errorless heuristic $\mathcal{R}$ *ε-solves f with zero error* with respect to input distribution μ if $\Pr_{x\sim\mu}[\mathcal{R}(x) = f(x)] \geq 1-\varepsilon$. Let $\mathrm{Suc}^{0\text{-err}}_{T,\mu}(f) := 1-\varepsilon$, where ε is the minimal value such that some T-query errorless heuristic ε-solves f with zero error with respect to μ. Note that $\mathrm{Suc}^{0\text{-err}}_{T,\mu}(f)$ is exactly $\mathrm{Suc}_{T,\mu}(V_f)$, where the search problem V_f is defined as

$$V_f := \{u \in \{0,1,*\}^n : u \text{ forces the value of } f\}.$$

Also, note that $V_{f^{\otimes k}} = V_f^{\otimes k}$. Thus, the following result is immediately implied by Theorem 8.5:

THEOREM 8.6. Suppose $\mathrm{Suc}^{0\text{-err}}_{T,\mu}(f) \leq 1-\varepsilon$. Then for $\alpha \in (0,1]$,

$$\mathrm{Suc}^{0\text{-err}}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(f^{\otimes k}) \leq \Pr_{Y \sim B_{k,1-\varepsilon}}[Y > (1-\alpha\varepsilon)k]\,.$$

Let us revisit the XOR problem in the current setting. It is easy to see that an errorless heuristic to compute the k-fold XOR $f^{\oplus k}$, on inputs drawn from a product distribution, cannot produce any output other than " ? " unless its queries allow it to determine the value of $f^{\otimes k}$. Thus, Theorem 8.6 also implies an XOR lemma with the same success bound for errorless heuristics.

Next, we prove a worst-case analogue of Theorem 8.6. Define $R_0(f)$, the *zero-error randomized query complexity of* f, as the minimum T for which some algorithm $\mathcal{R}$ outputs $f(x)$ with probability 1 for each x, and for which the *expected* number of queries made by $\mathcal{R}$ to any input is at most T. The following is another variant of Yao's minimax principle (Yao 1977); we include a proof for completeness.

LEMMA 8.7. Let $\eta \in (0,1]$. There exists a distribution μ_η over inputs to f, such that $\mathrm{Suc}^{0\text{-err}}_{\eta R_0(f),\mu_\eta}(f) \leq \eta$.

PROOF. Consider the following 2-player game: Player 1 chooses a (possibly randomized) errorless heuristic $\mathcal{R}$ for f which makes at most $\eta R_0(f)$ queries, and player 2 chooses (simultaneously) an input x to f. Player 1 wins if $\mathcal{R}(x) = f(x)$. We claim that there exists a randomized strategy for player 2, that is, a distribution $\mu =: \mu_\eta$ over inputs to x, that beats any strategy of player 1 with probability at least $1-\eta$. This will prove the Lemma.

To prove the claim, suppose for contradiction's sake that no such strategy for player 2 exists. Then, by the minimax theorem, there exists a randomized strategy for player 1 which wins with probability greater than η against all choices of x. This strategy is itself a randomized algorithm making at most $\eta R_0(f)$ queries; let us call this algorithm $\mathcal{R}$. Consider the algorithm $\mathcal{R}'$ for f that on input x, repeatedly applies $\mathcal{R}$ to x until $\mathcal{R}$ produces an output, which $\mathcal{R}'$ then outputs. We have $\mathcal{R}'(x) = f(x)$ on every input. Also, the expected number of queries of $\mathcal{R}'$ on any input is strictly less than

$$\sum_{m\geq 1}(1-\eta)^{m-1}\eta\left(m\cdot\eta R_0(f)\right) = \left(\sum_{m\geq 1}(1-\eta)^{m-1}m\right)\cdot\eta^2 R_0(f)$$
$$= \frac{1}{\eta^2}\cdot\eta^2 R_0(f)$$
$$= R_0(f),$$

contradicting the definition of $R_0(f)$. □

THEOREM 8.8. For any (not necessarily Boolean) function f, and $\alpha \in (0, 1/2]$, any errorless heuristic for $f^{\otimes k}$ using at most $\alpha^2 R_0(f)k/4$ queries has worst-case success probability less than $(7\alpha\ln(1/\alpha))^k$.

PROOF. Set $\gamma := \alpha/2$. Let μ_γ be the distribution given by Lemma 8.7, so that $\mathrm{Suc}^{0\text{-err}}_{\gamma R_0(f),\mu_\gamma}(f) \leq \gamma$. By Theorem 8.6 applied to α, with $T := \gamma R_0(f)$ and $\varepsilon := 1-\gamma$,

$$\mathrm{Suc}^{0\text{-err}}_{\alpha(1-\gamma)\gamma R_0(f)k,\mu_\delta^{\otimes k}}(f^{\otimes k}) \leq \Pr_{Y\sim B_{k,\gamma}}\left[Y > (1-\alpha(1-\gamma))k\right].$$

We have $\alpha^2 R_0(f)k/4 \leq \alpha(1-\gamma)\gamma R_0(f)k$ (using $\gamma \leq 1/2$), so that

$$\mathrm{Suc}^{0\text{-err}}_{\alpha^2 R_0(f)k/4,\mu_\gamma^{\otimes k}}(f^{\otimes k}) \leq \Pr_{Y\sim B_{k,\gamma}}\left[Y > (1-\alpha(1-\gamma))k\right]$$
$$< \left[1-(1-\gamma)+6\alpha\ln(1/\alpha)(1-\gamma))\right]^k$$

(applying Lemma 2.2, with $\beta := \alpha \leq 1/2$ and $\delta := (1-\gamma)$)

$$< (\alpha/2 + 6\alpha\ln(1/\alpha))^k$$
$$< (7\alpha\ln(1/\alpha))^k.$$ □

9. A direct product theorem for decision tree size

We measure the *size* of a decision tree $\mathcal{D}$, denoted size($\mathcal{D}$), as the number of leaf (output) vertices. Note that this is at least 1/2 the

total number of vertices. Define $\mathrm{Suc}^{\mathrm{size}}_{T,\mu}(f)$ as the maximum success probability of any size-T decision tree attempting to compute f on an input drawn from distribution μ. We have the following DPT for size-bounded query algorithms:

THEOREM 9.1. Let f be a Boolean function. Suppose $\mathrm{Suc}^{\mathrm{size}}_{T,\mu}(f) \leq 1-\varepsilon$. Then for $0 < \alpha \leq 1$, $\mathrm{Suc}^{\mathrm{size}}_{T^{\alpha\varepsilon k},\mu^{\otimes k}}(f^{\otimes k}) \leq 2^{\alpha\varepsilon k}(1-\varepsilon)^k$.

Note how the size bound grows exponentially, rather than linearly, in k in the above statement. It is natural to expect such a statement, since the k-fold application of a size-T decision tree is described by a size-T^k decision tree. Also note that, by convexity, Theorem 9.1 also bounds the success probability of any "randomized size-$T^{\alpha\varepsilon k}$ algorithm" $\mathcal{R}$, that is, of any probability distribution over size-$T^{\alpha\varepsilon k}$ decision trees.

PROOF. The proof follows that of Theorem 1.1, except that we need a new way to quantify the resources used by each of the k inputs. First, we develop some definitions pertaining to a single input to f. Given $u \in \{0,1,*\}^n$ and a real number $Z \in [1,T]$, let

$$W_{\mathrm{size}}(u,Z) := \mathrm{Suc}^{\mathrm{size}}_{Z,\mu^{(u)}}(f).$$

LEMMA 9.2. 1. $W_{\mathrm{size}}(*^n,T) \leq 1-\varepsilon$.

2. Take any real numbers $S^{(0)}, S^{(1)} \geq 1$ and let $S := S^{(0)} + S^{(1)}$. Then for any $u \in \{0,1,*\}^n$ and any $i \in [n]$,

$$\mathbb{E}_{\mathbf{y}\sim\mu^{(u)}}[W_{\mathrm{size}}(u[x_i \leftarrow \mathbf{y}_i], S^{(\mathbf{y}_i)})] \leq W_{\mathrm{size}}(u,S).$$

The proof is very similar to that of Lemma 3.1 and is omitted.

Now let $\mathcal{D}$ be any deterministic algorithm of size at most $T^{\alpha\varepsilon k}$ attempting to compute $f^{\otimes k}$ on input strings $\mathbf{x} = (\mathbf{x}^1,\ldots,\mathbf{x}^k) \sim \mu^{\otimes k}$. Let $M := \lfloor T^{\alpha\varepsilon k} \rfloor$; $\mathcal{D}$ always makes at most M queries.

As in previous proofs, for $j \in [k]$ and $0 \leq t \leq M$, let $u^j_t \in \{0,1,*\}^n$ describe the outcomes of all queries made to $\mathbf{x}^j$ after $\mathcal{D}$ has taken t steps (here a "step" consists of a query, unless $\mathcal{D}$ has halted, in which case a step has no effect).

Let S_t be defined as the size (number of leaf vertices) of the subtree of $\mathcal{D}$ reached after t steps have been taken. Thus, we have $S_0 \leq T^{\alpha\varepsilon k}$, and $S_t = 1$ if $\mathcal{D}$ has halted after at most t queries. For each $j \in [k]$, we define a sequence $Z_{j,0}, \ldots, Z_{j,M}$, as follows. Let $Z_{j,0} := T$. For $0 \leq t < M$, if $\mathcal{D}$ has halted after t steps, let $Z_{j,t+1} := Z_{j,t}$. Otherwise, if the $(t+1)$st query made by $\mathcal{D}$ is *not* to $\mathbf{x}^j$, we again let $Z_{j,t+1} := Z_{j,t}$. If the $(t+1)$st query is to $\mathbf{x}^j$, let

$$Z_{j,t+1} := \frac{S_{t+1}}{S_t} \cdot Z_{j,t}.$$

Let $X_{j,t} := W_{\text{size}}(u_j^t, Z_{j,t})$ if $Z_{j,t} \geq 1$; otherwise let $X_{j,t} := 1/2$. Let $P_t := \prod_{j\in[k]} X_{j,t}$. Arguing as in Theorem 1.1, for each $0 \leq t < M$, $\mathbb{E}[P_{t+1}] \leq \mathbb{E}[P_t]$. It follows that $\mathbb{E}[P_M] \leq \mathbb{E}[P_0] = W_{\text{size}}(*^n, T)^k \leq (1-\varepsilon)^k$.

Condition on any complete execution of $\mathcal{D}$, as described by $u_M^1, \ldots, u_M^k$. Notice that if $Z_{j,M} \geq 1$, then (by the definitions) $X_{j,M}$ is an upper bound on the conditional success probability of guessing $f(\mathbf{x}^j)$ correctly. Also, $X_{j,t} \geq 1/2$ for all j, t, and all inputs are independent after our conditioning. Thus, the conditional success probability of computing $f^{\otimes k}(\mathbf{x})$ is at most $2^{|B|} \cdot P_M$, where we define the (random) set $B := \{j \in [k] : Z_{j,M} < 1\}$.

Observe that $S_M = 1$, since the algorithm halts after at most M steps. Then,

$$\begin{aligned} 1 = S_M &= \left(\frac{S_1}{S_0}\right) \cdot \cdots \cdot \left(\frac{S_M}{S_{M-1}}\right) \cdot S_0 \\ &\leq \left(\frac{\prod_{j\in[k]} Z_{j,M}}{T^k}\right) \cdot T^{\alpha\varepsilon k} \\ &\leq T^{-|B|} \cdot T^{\alpha\varepsilon k}. \end{aligned}$$

Thus, $|B| \leq \alpha\varepsilon k$ always. So the *overall* success probability is at most $\mathbb{E}[2^{|B|} P_M] \leq 2^{\alpha\varepsilon k}\mathbb{E}[P_M] \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k$. □

One can also prove variants of our XOR lemma and other results in which we impose bounds on decision tree size rather than number of queries. We omit the details.

10. DPTs for dynamic interaction

So far, all of the computational tasks we have studied have involved algorithms querying a collection of fixed input strings. However, in many situations in computer science, it is natural to consider more general problems of *interaction* with dynamic, stateful entities. An algorithm can still "query" these entities, but these actions may influence the outcomes of future queries. In this section, we describe how our proof methods can yield DPTs for these more general problems. The methods involved are essentially the same as in previous sections, and the theorem we give is just one example of the kind of DPT we can prove for dynamic interaction, so we will only sketch the proofs here, indicating the novel elements.

We will propose a self-contained model of dynamic interaction. We make no claims of conceptual novelty for this model, however. Dynamic interaction has been an important concept for cryptography; in this context, Maurer (2002) proposed a model of *random systems* that generalizes our model. All of our work in this section could in principle be carried out in the random systems framework; we choose to use a different model that is somewhat simpler and adequate to our needs and that preserves a clear resemblance to our work in previous sections.

Much of the work in the random systems framework studies various kinds of *composition* of random systems; this work aims to understand how cryptographic primitives can be combined into more complex protocols. In this vein, Maurer *et al.* (2007) proved a result (see their Lemma 6) that can be informally described as follows: If an agent is playing games with two or more independent, noncommunicating entities, then the maximum joint-success probability is achieved by following independent strategies on the different games. This result establishes an "ideal" direct product property for interaction tasks with k independent entities, in which the number of queries to each entity is fixed in advance. By contrast, our focus will be on proving DPTs for query algorithms that can adaptively reallocate queries between the k entities.

Now we formally define the type of entity with which our query algorithms interact. Define an *interactive automaton (IA)* as a 5-tuple

$$\mathcal{M} = (\mathbf{seeds}, \mathbf{states}, \mathbf{queries}, R, \Delta), \text{ where:}$$

- **seeds**, **states**, **queries** are each finite sets, and **states** contains a distinguished *start-state* s_0;
- $R : \mathbf{seeds} \times \mathbf{states} \times \mathbf{queries} \to \{0,1\}$ is a *response mapping*;
- $\Delta : \mathbf{seeds} \times \mathbf{states} \times \mathbf{queries} \to \mathbf{states}$ is a *transition mapping*.

These automata are deterministic, but we can incorporate randomness by providing random bits as part of **seeds**.

We consider the scenario in which $\mathcal{M}$ is initialized to some seed $z \in \mathbf{seeds}$ according to a distribution μ, along with the start-state s_0. The automaton retains the value z throughout an interaction with a query algorithm $\mathcal{R}$ (which does not know the value z), but changes its state-value. If $\mathcal{R}$ selects the query $q \in Q$ while $\mathcal{M}$ has internal state $(z, s) \in \mathbf{seeds} \times \mathbf{states}$, then $\mathcal{M}$ returns the value $R(z, s, q)$ to $\mathcal{R}$ and transitions to the state $(z, \Delta(z, s, q))$.[7]

There are several kinds of tasks one can associate with an IA. One such task for the query algorithm $\mathcal{R}$ is to try to output a value $b \in B$ that satisfies some predicate $P(z, b)$, where z is the seed to $\mathcal{M}$ and $P \subseteq \mathbf{seeds} \times B$ is a total relation over **seeds** and a finite set B. This, of course, is a generalization of the relation problems we studied in Section 6, and it is natural to study the k-fold setting, in which $\mathcal{R}$ interacts with k IAs, querying one of them at each step. We assume that each IA only updates its state or sends a response to $\mathcal{R}$ when it is queried. In particular, the IAs do not communicate with each other.

We can transform the IA interaction scenario into an equivalent one, which highlights the similarity with the standard query model and makes it easy to apply our previous work to obtain a DPT. For simplicity assume $|\,\mathbf{seeds}\,| = 2^m$. Given an IA $\mathcal{M}$ and an integer $N > 0$, for each $z \in \mathbf{seeds}$, we define a string

[7] We can now sketch the modeling differences between our work and Maurer (2002). Maurer's "random systems" are modeled as inherently randomized; they may or may not be finite-state machines; and they are specified "behaviorally" by their conditional distributions over query responses, conditioned on all possible conversation transcripts.

$\xi(z) \in \{0,1\}^{m+(|\,\mathbf{queries}\,|+1)^N}$. There are two types of entries in this string. First there are m "ID" entries, which simply contain a binary encoding of z. Next, there are $(|\,\mathbf{queries}\,|+1)^N$ "response" entries, with each such entry indexed by an N-tuple $\overline{q} = (q_1, \ldots, q_N) \in (\mathbf{queries} \cup \{*\})^N$. We are only interested in response-entries of form $\overline{q} = (q_1, \ldots, q_r, *, *, \ldots, *)$, where $q_1, \ldots, q_r \in \mathbf{queries}$. For such an entry, we define $\xi(z)_{\overline{q}} \in \{0,1\}$ as the result of the following experiment: initialize $\mathcal{M}$ to state (z, s_0), and perform the interaction in which a query algorithm asks queries $q_1, \ldots, q_r$ in that order. Let $\xi(z)_{\overline{q}}$ be the final, rth response made by $\mathcal{M}$.

Define a total relation $P_\xi \subseteq \{0,1\}^{m+(|\,\mathbf{queries}\,|+1)^N} \times B$ by

$$P_\xi := \{(\xi(z), b) : z \in \mathbf{seeds} \wedge P(z,b)\}.$$

Also, given a distribution μ over **seeds**, define $\mu_\xi \sim \xi(z)$, where $z \sim \mu$. In this way, we map an IA interaction task onto a relation problem of the type studied in Section 6, with a corresponding map from initialization distributions to input distributions.

A standard query algorithm $\mathcal{R}$ (as studied in all previous sections) can faithfully simulate an interaction with $\mathcal{M}$ initialized to an unknown $z \in \mathbf{seeds}$, if given query access to $\xi(z)$. This works in the natural way: If its simulated queries up to the rth step are $q_1, \ldots, q_r$, then for its rth query to $\xi(z)$, $\mathcal{R}$ looks at the entry $(q_1, \ldots, q_r, *, *, \ldots, *)$ to learn $\mathcal{M}$'s rth response. Call an algorithm "interaction-faithful" if its sequence of queries to any input string always obeys this format.

Of course, not all algorithms are interaction-faithful. For example, an unfaithful algorithm could simply look at the ID-entries to learn z. Thus, the relation problem (P_ξ, μ_ξ) can be much easier than the IA interaction problem defined by $(\mathcal{M}, P, \mu)$. However, if we restrict attention to the class of interaction-faithful algorithms $\mathcal{R}$, then it is not hard to see that there is an *exact* correspondence between the "difficulty" of the two problems, at least for interactions lasting at most N steps. That is, for $T \leq N$, there is a T-query IA interaction algorithm for $(\mathcal{M}, P, \mu)$ with success probability p, if and only if there is a T-query interaction-faithful standard algorithm for (P_ξ, μ_ξ) with success probability p.

The good news is that we can prove a DPT for interaction-faithful query algorithms in almost exactly the same way as for unrestricted query algorithms. In fact, it is most natural to prove a DPT for a more general notion of faithfulness, which we define next. Say we are given $n > 0$ and a map $\tau : \{0,1,*\}^n \to \{0,1\}^n$, called a *query-restriction map*. Say that a (standard) query algorithm $\mathcal{R}$ on n input bits is *τ-faithful* if for every execution of $\mathcal{R}$ on any input, whenever the input bits seen by $\mathcal{R}$ seen so far are given by $u \in \{0,1,*\}^n$, then $\mathcal{R}$ either halts or chooses a next input bit x_i to query whose index satisfies $\tau(u)_i = 1$. In other words, a restriction map τ restricts the possible next queries that can be made by a τ-faithful algorithm, in a way that depends only on the description u of the bits seen so far. Note that interaction-faithfulness as defined earlier is indeed equivalent to τ-faithfulness for an appropriately defined $\tau = \tau_{\text{int}}$.

For $k > 1$, define the *k-fold product* of restriction map τ, denoted $\tau^{\otimes k} : \{0,1\}^{kn} \to \{0,1\}^{kn}$, by $\tau^{\otimes k}(u^1, \ldots, u^k) := (\tau(u^1), \ldots, \tau(u^k))$. The map $\tau^{\otimes k}$ can be interpreted as a restriction map for algorithms making queries to a collection $x^1, \ldots, x^k$ of n-bit strings. Note that $\mathcal{R}$ is $\tau^{\otimes k}$-faithful exactly if for each $j \in [k]$, $\mathcal{R}$'s queries to the jth input (considered alone) are always τ-faithful. Thus, the k-fold IA interaction problem defined by $(\mathcal{M}, P, \mu)$ has "difficulty" equivalent to the k-fold relation problem defined by (P_ξ, μ_ξ) for $\tau_{\text{int}}^{\otimes k}$-faithful algorithms, provided N is chosen large enough in the definition of $\xi(\cdot)$ (relative to the query bounds we are interested in).

In light of these observations, a DPT for IA interaction algorithms follows by straightforward translation from the following DPT (generalizing Theorem 6.3) for standard query algorithms obeying a restriction map:

THEOREM 10.1. Let $P \subseteq \{0,1\}^n \times B$ be a total relation such that any T-query, τ-faithful algorithm solves P with probability at most $1 - \varepsilon$ under input distribution μ.

For any algorithm $\mathcal{R}$ making queries to inputs $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$ and producing output in B^k, define the random set $S[\mathbf{x}]$ as in Theorem 6.3.

Suppose $\mathcal{R}$ is $\tau^{\otimes k}$-faithful and $\alpha\varepsilon Tk$-query-bounded for some $\alpha \in (0,1]$, and $\mathcal{A}$ is any monotone subset of $\mathcal{P}([k])$. Then, conclusions 1 and 2 in Theorem 6.3 also hold for $\mathcal{R}$.

Proof. (Sketch) The proof follows that of Theorem 6.3; we only describe the differences. For $u \in \{0,1,*\}^n$, and for a deterministic algorithm $\mathcal{D}$ on n input bits, let

$$W_P(u,\mathcal{D}) := \Pr_{\mathbf{y}\sim\mu^{(u)}}[(\mathbf{y},\mathcal{D}(\mathbf{y})) \in P].$$

Let us say that $\mathcal{D}$ is *u-inducing* if, on any input $x \in \{0,1\}^n$ which extends[8] u, the outcome of $\mathcal{D}$'s first $|u|$ queries to x are described by u.

If $|u| \leq T$, define $W_{P,\tau}(u) := \max_{\mathcal{D}} W_P(u,\mathcal{D})$, where the max ranges over all deterministic, u-inducing, τ-faithful algorithms $\mathcal{D}$ making at most T queries. We have:

Lemma 10.2. 1. $W_{P,\tau}(*^n) \leq 1-\varepsilon$.

2. For any $u \in \{0,1,*\}^n$ with $|u| < T$, and any $i \in [n]$ satisfying $\tau(u)_i = 1$, we have

$$\mathbb{E}_{\mathbf{y}\sim\mu^{(u)}}[W_{P,\tau}(u[x_i \leftarrow \mathbf{y}_i])] \leq W_{P,\tau}(u).$$

The proof of Lemma 10.2 follows that of Lemma 3.1. The rest of the proof of Theorem 10.1 follows that of Theorem 6.3, with $W_{P,\tau}(u)$ taking the place of $W_P(u)$. □

One can also prove a DPT for search problems for τ-faithful query algorithms, along the lines of Theorem 8.1. When applied to interactive automata via the translation described earlier, search problems correspond to tasks whose success conditions are defined in terms of the interaction itself (rather than the hidden seed of the IA, or any output produced by the query algorithm).

11. Questions for future work

1. Can the bounds in our threshold DPTs and XOR lemma be improved? For example, in Theorem 1.3, can one improve the success probability bound to $\frac{1}{2}\left(1+[1-2\varepsilon+O(\alpha\varepsilon)]^k\right)$?

[8] (As defined in Section 8.1).

2. It is still unknown what worst-case success probability in computing $f^{\otimes k}$ can be achieved in general, when the number of queries allowed is $\alpha R_2(f)k$ for $\alpha \gg 1$. The corresponding question in the quantum query model was settled by Buhrman *et al.* (2007). As mentioned earlier, $O(R_2(f)k \log k)$ queries always suffice to compute $f^{\otimes k}$ with high success probability; the work of Feige *et al.* (1994) implies that we cannot do better than this by using a bounded-error randomized algorithm for f in a black-box fashion.

3. Can ideas from our work be helpful in obtaining new results in other computational models? For example, Lee & Roland (2011) prove a threshold DPT for quantum query algorithms computing Boolean functions, where the query bound scales as $\Omega(Q_2(f)k)$. Can we extend this to a *generalized* threshold DPT, analogous to our Theorem 6.3?

Acknowledgements

I thank Ronald de Wolf for numerous helpful comments, and in particular for encouraging me to look at threshold DPTs. I also thank the anonymous reviewers. This research was conducted with support from a DARPA YFA grant of Scott Aaronson.

References

Andris Ambainis, Loïck Magnin, Martin Roetteler & Jérémie Roland (2011). Symmetry-Assisted Adversaries for Quantum State Generation. In *26th IEEE Conference on Computational Complexity*, 167–177.

Andris Ambainis, Robert Špalek & Ronald de Wolf (2009). A New Quantum Lower Bound Method, with Applications to Direct Product Theorems and Time-Space Tradeoffs. *Algorithmica* **55**(3), 422–461. Earlier version in STOC 2006.

Harry Buhrman, Ilan Newman, Hein Röhrig & Ronald de Wolf (2007). Robust Polynomials and Quantum Algorithms. *Theory Comput. Syst.* **40**(4), 379–395. Earlier version in STACS 2005.

Harry Buhrman & Ronald de Wolf (2002). Complexity Measures and Decision Tree Complexity: a Survey. *Theor. Comput. Sci.* **288**(1), 21–43.

Devdatt P. Dubhashi & Alessandro Panconesi (2009). *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition.

Uriel Feige, Prabhakar Raghavan, David Peleg & Eli Upfal (1994). Computing with Noisy Information. *SIAM J. Comput.* **23**(5), 1001–1018. Earlier version in STOC 1990.

Oded Goldreich, Noam Nisan & Avi Wigderson (1995). On Yao's XOR-Lemma. *Electronic Colloquium on Computational Complexity (ECCC)* **TR95-050**.

Thomas Holenstein (2005). Key Agreement from Weak Bit Agreement. In *37th ACM STOC*, 664–673.

Thomas Holenstein & Grant Schoenebeck (2011). General Hardness Amplification of Predicates and Puzzles - (Extended Abstract). In *TCC*, 19–36.

Russell Impagliazzo (1995). Hard-Core Distributions for Somewhat Hard Problems. In *36th IEEE FOCS*, 538–545.

Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets & Avi Wigderson (2010). Uniform Direct Product Theorems: Simplified, Optimized, and Derandomized. *SIAM J. Comput.* **39**(4), 1637–1665. Earlier version in STOC 2008.

Russell Impagliazzo & Valentine Kabanets (2010). Constructive Proofs of Concentration Bounds. In *14th APPROX-RANDOM*, 617–631.

Russell Impagliazzo, Ran Raz & Avi Wigderson (1994). A Direct Product Theorem. In *9th IEEE Structure in Complexity Theory Conference*, 88–96.

Russell Impagliazzo & Avi Wigderson (1997). $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *29th ACM STOC*, 220–229.

Rahul Jain (2011). New strong direct product results in communication complexity. *Electronic Colloquium on Computational Complexity (ECCC)* **TR11-024**.

Rahul Jain, Hartmut Klauck & Miklos Santha (2010). Optimal Direct Sum Results for Deterministic and Randomized Decision Tree Complexity. *Inf. Process. Lett.* **110**, 893–897.

Rahul Jain, Attila Pereszlényi & Penghui Yao (2012). A direct product theorem for bounded-round public-coin randomized communication complexity. *CoRR* **abs/1201.1666**.

Hartmut Klauck (2010). A Strong Direct Product Theorem for Disjointness. In *42nd ACM STOC*, 77–86.

Hartmut Klauck, Robert Špalek & Ronald de Wolf (2007). Quantum and Classical Strong Direct Product Theorems and Optimal Time-Space Tradeoffs. *SIAM J. Comput.* **36**(5), 1472–1493. Earlier version in FOCS 2004.

Troy Lee & Jérémie Roland (2011). A Strong Direct Product Theorem for Quantum Query Complexity. *CoRR* **abs/1104.4468**. To appear in CCC 2012.

Troy Lee, Adi Shraibman & Robert Špalek (2008). A Direct Product Theorem for Discrepancy. In *23rd IEEE Conference on Computational Complexity*, 71–80.

Ueli M. Maurer (2002). Indistinguishability of Random Systems. In *21st EUROCRYPT*, 110–132.

Ueli M. Maurer, Krzysztof Pietrzak & Renato Renner (2007). Indistinguishability Amplification. In *27th CRYPTO*, 130–149.

Noam Nisan (1991). CREW PRAMs and Decision Trees. *SIAM J. Comput.* **20**(6), 999–1007. Earlier version in STOC 1989.

Noam Nisan, Steven Rudich & Michael E. Saks (1999). Products and Help Bits in Decision Trees. *SIAM J. Comput.* **28**(3), 1035–1050. Earlier version in FOCS 1994.

Ronen Shaltiel (2003). Towards Proving Strong Direct Product Theorems. *Computational Complexity* **12**(1-2), 1–22. Earlier version in CCC 2001.

Alexander A. Sherstov (2011). Strong direct product theorems for quantum communication and query complexity. In *43rd ACM STOC*, 41–50.

Robert Špalek (2008). The Multiplicative Quantum Adversary. In *23rd IEEE Conference on Computational Complexity*, 237–248.

Falk Unger (2009). A Probabilistic Inequality with Applications to Threshold Direct-Product Theorems. In *50th IEEE FOCS*, 221–229.

Emanuele Viola & Avi Wigderson (2008). Norms, XOR Lemmas, and Lower Bounds for Polynomials and Protocols. *Theory of Computing* **4**(1), 137–168. Earlier version in CCC 2007.

Andrew Chi-Chih Yao (1977). Probabilistic Computations: Toward a Unified Measure of Complexity (Extended Abstract). In *18th IEEE FOCS*, 222–227.

Manuscript received 7 June 2011

Andrew Drucker
Massachusetts Institute of
Technology,
Cambridge, MA, USA.
adrucker@mit.edu