Electronic Colloquium on Computational Complexity, Report No. 80 (2010)

Improved Direct Product Theorems for Randomized Query Complexity

Andrew Drucker*

May 4, 2010

Abstract

The *direct product problem* is a fundamental question in complexity theory which seeks to understand how the difficulty of 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 tradeoff between the query bound and the error probability. As a corollary, 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 $O(bs(f)k)$.

The proof 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 dyamic entities. We also give a version of our DPT in which decision tree size is the resource of interest.

*MIT, EECS Dept., Cambridge, MA, USA. Email: adrucker@mit.edu. Supported by a DARPA YFA grant.

ISSN 1433-8092

1 Introduction

1.1 Direct product problems

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 there are more chances to make a mistake. One way to make this idea more precise is the so-called *direct product problem*. In this approach, 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 which 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 [JKS10].

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' $f^{\oplus k}(x^1, \ldots, x^k) := f(x^1) \oplus \ldots \oplus f(x^k)$ of k independent inputs to f. An *XOR lemma* is a result which 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 [Ung09, IK10] for an overview and recent results of this type).

The direct product problem has been studied extensively in models including Boolean circuits (e.g., [GNW95, IW97, IJKW10]), communication protocols [IRW94, Sha03, KŠdW07, LSŠ08, VW08], and query algorithms [IRW94, NRS99, Sha03, KŠdW07]. In all of these models, an optimal T-bounded algorithm which attempts to compute f can always be applied independently to

[1] Terminology varies somewhat in the literature. For instance, what we call XOR lemmas are called 'direct product theorems' in [Sha03], and what we refer to as direct product problems are in [Sha03] called the 'concatenation variant'.

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 [Sha03], 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]

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 [Sha03] 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 by Nisan, Rudich, and Saks [NRS99].

Second, DPTs have been shown for unrestricted algorithms, but using resource bounds whose strength depends on properties of the function f. For example, Klauck, Špalek, and de Wolf [KŠdW07] 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 [Nis89, BdW02], 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, $bs(f) = \Theta(R_2(f))$ and so the DPT of [KŠdW07] gives good results. However, examples are known [BdW02] 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 success probability p' drops exponentially in k if the number of allowed quantum queries is $O(\sqrt{bs(f)}k)$. Špalek [Š08] proved a DPT for quantum query algorithms where the resource bound T' scales in terms of a complexity measure called the *multiplicative quantum adversary*. The ultimate strength of this result is not yet clear since the relationships between the multiplicative adversary and other complexity measures are not well-understood. Finally, for symmetric functions, direct product theorems of a strong form were proved for quantum query complexity by Ambainis, Špalek, and de Wolf [ASdW09].

We have surveyed results in the query model, but in other models, such as communication complexity, our earlier remarks also apply: known direct product theorems in which the allowed communication T' scales with k either apply to specific functions (e.g., in [KŠdW07] a direct product theorem was proved for the quantum communication complexity of the Disjointness function, and a classical analogue was proved by Klauck [Kla09]), or else require the allowed resources T' to scale as $\mathcal{D}(f)k$, where $\mathcal{D}(f)$ is a complexity measure which can be significantly smaller than the resources needed to compute a single instance of f. For example, in communication complexity, DPTs have been shown whose strength is related to the so-called *discrepancy* of f [Sha03, LSŠ08]. 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

[2] Shaltiel calls a DPT 'strong' if it applies to all p, T and its parameters satisfy $p' \leq p^{\Omega(k)}$ and $T' \geq \Omega(Tk)$. His counterexamples rule out strong DPTs for 'reasonable' computational models. In later works, the modifier 'strong' has been used in a somewhat broader way. We will not use this terminology in the present paper.

[3] Technically, 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.

p' decay as k grows (although if T' is allowed to shrink with k, a DPT with $p' = p^k$ can be shown using [Imp95, Hol05], as remarked in [IK10]).

1.2 Our results

Our main result is the following direct product theorem in the average-case setting:

Theorem 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 $\alpha \in (0,1]$, 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 tradeoff in Theorem 1 between the query bound and the error probability is essentially best-possible, at least for general functions f and for small values $\alpha < .001$. (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 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.

As a corollary of Theorem 1, we obtain the following DPT for worst-case error, which strengthens the worst-case DPT of [KŠdW07] mentioned earlier:

Theorem 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 a statement like Theorem 2 should be true, and proving such a DPT was arguably one of the major open problems in classical query complexity.

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

Theorem 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 $\frac{1}{2}\left(1 + [1-2\varepsilon+21\alpha\ln(2/\alpha)\varepsilon]^k\right)$.

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 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), we also prove a 'threshold' DPT, which bounds the probability that a query-bounded algorithm for $f^{\otimes k}$ solves 'too many' of the k instances correctly. As one special case, we prove:

Theorem 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 sampled from μ. Fix any $\eta, \alpha \in (0,1]$. Consider any randomized algorithm $\mathcal{R}$ making at most $\alpha\varepsilon Tk$ queries on k independent inputs from μ. Then 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 4 gives success bounds which 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 4, we also get an ordinary DPT for non-Boolean functions, which in general is stronger than the DPT we'd get by a straightforward generalization of our techniques for Theorem 1. This is the simplest way we know to get such a DPT.

Threshold DPTs have been proved for a variety of models. Unger [Ung09] shows how to derive threshold DPTs from XOR lemmas, and recent work of Impagliazzo and Kabanets [IK10] gives a way to derive threshold DPTs from sufficiently strong DPTs (see also the earlier works cited in [Ung09, IK10]). However, the results of [IK10] do not apply for our purposes, and the threshold DPT we prove is more general than we'd get by applying the results of [Ung09] to our XOR lemma. In any case the proof of our threshold DPT is, we feel, quite natural (and actually forms the basis for our XOR lemma). Our method 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 and Schoenebeck [HS10] 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 relation problems (for which direct *sum* theorems were proved recently by Jain, Klauck, and Santha [JKS10]), learning tasks, search problems, and errorless heuristics, as well as a DPT in which decision tree *size*, rather than depth, is the resource of interest. A DPT for decision tree size was shown previously in [IRW94], which gave an 'ideal' success probability decay $p' = p^k$, but in the case where the size is not allowed to scale with k, i.e., 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.

In order to ease notation, in this paper we discuss only DPTs for total functions, but our results apply to partial functions (functions with a restricted domain) as well; 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 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. As mentioned earlier, Shaltiel [Sha03] proved an optimal DPT for 'fair' decision trees, in which each of the k inputs receives T queries. Our proof method for Theorem 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 $X_{j,t} \in [1/2, 1]$ which represents the gambler's 'fortune' at the j-th 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 j-th input. When $\mathcal{D}$ queries the j-th input, the j-th 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.[4] 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.

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 j-th input has been queried more than T times. 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 which 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.

Our worst-case DPT for Boolean functions is a straightforward corollary of Theorem 1. 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 exactly the same as in Theorem 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 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.[5]

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, this is just the success probability he would achieve if he followed

[4] 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.

[5] 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.

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 show by a simple induction 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, as well as our XOR lemma, will follow readily from this handy 'gambling lemma' (Lemma 9).

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 and 2. We will introduce other definitions as needed in later sections. In Section 3 we prove Theorem 1, and in Section 4 we use Shaltiel's examples to analyze the tightness of this result. We prove Theorem 2 in Section 5.

In Section 6 we prove our 'gambling lemma', Lemma 9, and use it to prove a generalized threshold DPT for relation problems. Theorem 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 3, our XOR lemma, in Section 7 (also using Lemma 9). We define search problems and errorless heuristics in Section 8, and give DPTs for these settings.

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. We let $\mathrm{supp}(X)$ denote the support of a random variable X, i.e., 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 (deterministic) *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 *2-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 randomized algorithm $\mathcal{R}$ ε-computes f with respect to μ. By standard arguments, this maximum exists (and is attained by a deterministic height-T decision tree).

For $f : \{0,1\}^n \to B$ and $k \geq 1$, define $f^{\otimes k} : \{0,1\}^{kn} \to \{0,1\}^k$, 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 $f^{\oplus k}(x^1, \ldots, x^k) := f(x^1) \oplus \ldots \oplus f(x^k)$, 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 form of Chernoff's inequality will be convenient for us. The proof is in Appendix A.

Lemma 5. *Let $\delta \in (0,1)$, and let $Y \sim B_{k,1-\delta}$. If $\beta \in (0, 1/2]$, then*

$$\Pr[Y > (1 - \beta\delta)k] < \left[1 - \delta + 21\beta \ln(1/\beta)\delta\right]^k.$$

To apply Lemma 5, 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 example, if $n > 1$ 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

In this section we prove our 'basic' direct product theorem:

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

Proof. First we make some simplifying observations. The statement is a triviality if $T = 0$ or $\varepsilon = 0$, so assume both are positive. Also, by convexity, it is sufficient to show the statement for deterministic algorithms. Finally, 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; afterwards 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 i-th coordinate set to b. In our proof we consider the bits of an input $\boldsymbol{y} \sim \mu$ 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 $\Pr_{\boldsymbol{y}\sim\mu^{(u)}}[\boldsymbol{y}_i = 0]$, 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 to regarding the input as being drawn from μ before the algorithm begins (this is the 'principle of deferred decisions' of probability theory).

For each $u \in \{0,1,*\}^n$, and each deterministic algorithm $\mathcal{D}$ on n input bits, let $W(u,\mathcal{D}) := \Pr_{\boldsymbol{y}\sim\mu^{(u)}}[\mathcal{D}(\boldsymbol{y}) = f(\boldsymbol{y})]$. If $|u| \le T$, we define $W^\star(u) := \max_{\mathcal{D}} W(u,\mathcal{D})$, where the max ranges over all deterministic algorithms $\mathcal{D}$ making at most $(T - |u|)$ queries. Clearly $W^\star(u) \in [1/2, 1]$, since an algorithm may simply guess a random bit. We make two more simple claims about this function.

Lemma 6. *1. $W^\star(*^n) \le 1 - \varepsilon$.*

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

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

For 2: If the i-th coordinate has already been queried (i.e., $u_i \in \{0,1\}$), then $\boldsymbol{y}_i = u_i$ with probability 1, so $u[x_i \leftarrow \boldsymbol{y}_i] = u$ and the statement is trivial. So assume $u_i = *$. Let $\mathcal{D}_0$, $\mathcal{D}_1$ 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; that is, $W(u[x_i \leftarrow 0], \mathcal{D}_0) = W^\star(u[x_i \leftarrow 0])$ and $W(u[x_i \leftarrow 1], \mathcal{D}_1) = W^\star(u[x_i \leftarrow 1])$. Consider an algorithm $\mathcal{D}$ which queries x_i, then runs $\mathcal{D}_b$ if the bit seen is b. $\mathcal{D}$ makes at most $T - |u|$ queries, and we compute $W(u,\mathcal{D}) = \mathbb{E}_{\boldsymbol{y}\sim\mu^{(u)}}[W^\star(u[x_i \leftarrow \boldsymbol{y}_i])]$. Thus $W^\star(u)$ 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 $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$ sampled from $\mu^{\otimes k}$. For $j \in [k]$ and $0 \le t \le M$, let $u_t^j \in \{0,1,*\}^n$ be the random string giving the outcomes of all queries made to $\boldsymbol{x}^j$ after $\mathcal{D}$ has made t queries (to the entire input). We need the following important observation:

Lemma 7. *Condition on any execution of $\mathcal{D}$ for the first $t \ge 0$ steps, with query outcomes given by $u_t^1, \ldots, u_t^k$. Then the input is in the state $\mu^{(u_t^1)} \times \ldots \times \mu^{(u_t^k)}$; that is, the k inputs are independent, with $\boldsymbol{x}^j$ distributed as $\mu^{(u_t^j)}$.*

We include the simple proof in Appendix B.

Define collections

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

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 $X_{j,t} := W^\star(u_j^t)$ if $|u_j^t| \le T$; otherwise let $X_{j,t} := 1/2$. Let $P_t := \prod_{j\in[k]} X_{j,t}$.

We claim that for each $0 \le t < M$, $\mathbb{E}[P_{t+1}] \le \mathbb{E}[P_t]$. 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 i-th bit of the j-th 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 $P_{t+1} = P_t$. So assume $|u_j^t| < T$. Then we have

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

where we used part 2 of Lemma 6. 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: $P_0 = W^\star(*^n)^k \leq (1-\varepsilon)^k$ (Lemma 6, 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 $\mathcal{D}$. For $j \in [k]$, define the random variable $\Delta_j := X_{j,M} - 1/2$, so $\Delta_j \in [0, 1/2]$. Condition on the bits $u_M^1, \ldots, u_M^k$ seen by $\mathcal{D}$ during a complete execution. For each $j \in [k]$, there are two possibilities: either $|u_M^j| > T$, or the j-th input is in a final state $\mu^{(u_M^j)}$ for which $\Pr_{\boldsymbol{x} \sim \mu^{(u_M^j)}}[f(\boldsymbol{x}) = 1] \in [1/2 - \Delta_j, 1/2 + \Delta_j]$. Since the k inputs remain independent under our conditioning, the conditional probability that $\mathcal{D}$ computes $f^{\otimes k}$ correctly is at most $\prod_{j:|u_M^j| \leq T}(1/2 + \Delta_j) = \prod_{j:|u_M^j| \leq T} X_{j,M}$.

$\mathcal{D}$ 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,

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

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

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 Theorem follows. □

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

In this section we describe a family of functions and input distributions, due to Shaltiel [Sha03], and explain why they show that the query/success tradeoff in Theorem 1 is nearly best-possible, at least when $\alpha < .001$ 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 $f_T(x) := x_2 \oplus \ldots \oplus x_{T+2}$. 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 $\boldsymbol{y} \sim \mu_\varepsilon$, 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 $f(\boldsymbol{y})$.

Now consider the following algorithm $\mathcal{D}$ attempting to compute $f^{\otimes k}$ on inputs $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k \sim \mu_\varepsilon^{\otimes k}$. First $\mathcal{D}$ queries the first two bits of each input. Call an input $\boldsymbol{x}^k$ 'bad' if its first bit is 0, '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 $2k + \alpha\varepsilon T k$ queries total. To analyze the success probability of $\mathcal{D}$, first consider an algorithm $\mathcal{D}'$ which uses only $2k$ queries to look at the first index of each input, outputting the correct value on good inputs and guessing 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 $\mathcal{D}$ and $\mathcal{D}'$ are both 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 conditional success probability of $\mathcal{D}$ is $2^{\lfloor \alpha\varepsilon k \rfloor}$ times the conditional success probability of $\mathcal{D}'$, since $\mathcal{D}$ has $\lfloor \alpha\varepsilon k \rfloor$ fewer random guesses to make. Thus,

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

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 $Y := Y_1 + \ldots + Y_k$. We apply Lemma 5 to Y, with the settings $\delta := 2\varepsilon$ and $\beta := \alpha/2 \leq 1/2$, to obtain

$$\begin{aligned}
\Pr[|B| < \alpha\varepsilon k] &= \Pr[Y > (1 - \alpha\varepsilon)k] \\
&= \Pr[Y > (1 - (2\varepsilon)(\alpha/2))k] \\
&< \left[1 - 2\varepsilon + 21(\alpha/2)\ln(2/\alpha)(2\varepsilon)\right]^k.
\end{aligned}$$

This can be made less than $(1 - 1.5\varepsilon)^k$ if α is a small enough positive constant ($\alpha < .001$ 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. (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 if $\mathcal{D}$ used $\alpha\varepsilon T k$ queries (recall, though, that $\mathcal{D}$ uses $2k + \alpha\varepsilon T k$ queries).

5 Proof of Theorem 2

We now prove Theorem 2 from the Introduction, our DPT for worst-case error, by combining Theorem 1 with a version of Yao's minimax principle [Yao77], 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, which can be proved directly using the Minimax Theorem of game theory.

Lemma 8. *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 2. Let f be given. Let $\delta := 1/2 - \gamma/2$, and let $\mu := \mu_\delta$ be as provided by Lemma 8, so that every algorithm making fewer than $R_{2,\delta}(f)$ queries succeeds in computing f on μ with probability less than $1-\delta$. We apply Theorem 1 to f, μ, where we have $\varepsilon > \delta > 3/8$. With the setting $\alpha := \gamma$, we conclude that any algorithm making fewer than $\alpha\varepsilon R_{2,\delta}(f)k$ 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 $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k \sim \mu^{\otimes k}$. 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 which 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 i-th trial], and let $Y := Y_1 + \ldots + Y_m$. 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 $\gamma < 1/4$). Now

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

so any 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 $f^{\otimes k}$. □

6 Threshold Direct Product Theorems

In this section we prove our 'gambling lemma', Lemma 9, and use it to prove generalized threshold DPTs for relation problems (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 9 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 $A \in \mathcal{A}, A \subseteq B$ implies $B \in \mathcal{A}$. Monotone collections play an important role in what follows.

6.1 A gambling lemma

We now prove a technical result, Lemma 9 below, that will play a key role in the rest of the paper. Like the proof of Theorem 1, this Lemma's statement 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 j-th 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 j-th table, the gambler wins a fortune of 1 with probability p_j, and loses his j-th endowment with the remaining probability. The following Lemma states that this is in fact the gambler's best strategy.

Lemma 9. *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 j-th table in the first t steps). *$\mathcal{U} = \{U_0, U_1, \ldots, U_{M-1}\}$ 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 $0 \leq t' \leq t < N, j \in [k]$, $X_{j,t'}$ is determined by U_t, and that $\mathbb{E}[X_{j,t+1}|U_t] \leq X_{j,t}$. 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'll 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 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. To see this, consider any family $\mathcal{X}$ obeying the Lemma's assumptions, and 'split' each transition $(t, t+1)$ into a sequence of k transitions, in the j-th of which the j-th fortune is subject to change (according to the same distribution governing its transition between t and $t+1$ in the original sequence). The Lemma's assumptions continue to hold for this modified family of sequences; 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 assume

from now on that $\mathcal{X}$ obeys this extra assumption, 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 $S \cup \{j\} \notin \mathcal{A}$. In this case, $\Pr[\mathcal{A}\text{-success}] = 0$, and again the conclusion is trivially satisfied. So suppose $S \notin \mathcal{A}$, $S \cup \{j\} \in \mathcal{A}$, and condition on any value $U_0 = u$. Then $\mathcal{A}$-success occurs iff $X_{j_0,1} = 1$. 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 obey the Lemma'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

$$\Pr[\mathcal{A}\text{-success} | U_0 = u, X_{j_0,1} = a] \leq \Pr[D^{(a)} \in \mathcal{A}], \tag{2}$$

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. (2),

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

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. First we define relation problems. 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 $(x,b) \in P$. Of course, 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, and P is a total relation, say that $\mathcal{R}$ *ε-solves the relation problem P* if for all inputs x, $\Pr[(x, \mathcal{R}(x)) \in P] \geq 1 - \varepsilon$. Similarly, if μ is a distribution over inputs $x \in \{0,1\}^n$, we say that $\mathcal{R}$ *ε-solves the relation problem P with respect to μ* if $\Pr_{x \sim \mu}[(x, \mathcal{R}(x)) \in P] \geq 1 - \varepsilon$. Define $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) := 1 - \varepsilon$, where $\varepsilon \geq 0$ is the

minimum value for which some T-query randomized algorithm $\mathcal{R}$ ε-solves P with respect to μ. As usual, this maximum exists and is attained by a deterministic height-T decision tree. Given any randomized algorithm $\mathcal{R}$ making queries to $k \geq 1$ inputs $x = (x^1, \ldots, x^k)$ to the relation problem P and producing an output in B^k, let $\mathcal{R}_j(x) \in B$ be the j-th value outputted by $\mathcal{R}$.

Given $A, B \subseteq [k]$, define the *distance* $d(A,B) := |(A \setminus B) \cup (A \setminus B)|$. 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

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

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 are now prepared to state our generalized threshold DPT:

Theorem 10. *Let $P \subseteq \{0,1\}^n \times B$ be a total relation for which $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) \leq 1 - \varepsilon$. Fixing any randomized algorithm $\mathcal{R}$ making queries to inputs $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \sim \mu^{\otimes k}$ and producing output in B^k, define the (random) set*

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

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. *$\Pr[S[\boldsymbol{x}] \in \mathcal{A}] \leq |B|^{\alpha\varepsilon k} \cdot \Pr[D \in \mathcal{A}]$, 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, $\Pr[S[\boldsymbol{x}] \in \mathcal{A}] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]$.*

Proof. As in Theorem 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 $u \in \{0,1,*\}^n$, and for a deterministic algorithm $\mathcal{D}$ on n input bits, let $W_P(u, \mathcal{D}) := \Pr_{\boldsymbol{y} \sim \mu^{(u)}}[(\boldsymbol{y}, \mathcal{D}(\boldsymbol{y})) \in P]$. If $|u| \leq T$, define $W^\star_P(u) := \max_{\mathcal{D}} W_P(u, \mathcal{D})$, where the max ranges over all deterministic algorithms $\mathcal{D}$ making at most $(T - |u|)$ queries. Then $W^\star_P(u) \in [|B|^{-1}, 1]$. We have the following claim, whose proof follows that of Lemma 6:

Lemma 11. *1. $W^\star_P(*^n) \leq 1 - \varepsilon$.*

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

Now we prove the Theorem. Let $\mathcal{R}$ be given; as in Theorem 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^j_t defined in Theorem 1.

Define random variables $\{X_{j,t}\}_{j \in [k], 0 \leq t \leq M}$, determined by an execution of $\mathcal{D}$ on inputs $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \sim \mu^{\otimes k}$, by letting $X_{j,t} := W^\star_P(u^j_t)$ if $|u^j_t| \leq T$, otherwise $X_{j,t} := |B|^{-1}$. Next, the natural idea is to apply Lemma 9. First, however, we need to extend the sequences for one additional (non-query) 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 $|u^j_M| \leq T$, that is, on whether $\mathcal{D}$ made at most T queries to $\boldsymbol{x}^j$ on the current execution. If $|u^j_M| \leq T$, let $X_{j,M+1} := \mathbf{1}_{[(\boldsymbol{x}^j, \mathcal{D}_j(\boldsymbol{x})) \in P]}$ be the indicator variable for the event that $\mathcal{D}$ solves the relation problem on the j-th input. If $|u^j_M| > T$, 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 $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k)$.

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 9 are satisfied by $(\mathcal{X}, \mathcal{U})$, with $N := M + 1$. First, for $0 \leq t' \leq t \leq M$, $X_{j,t'}$ is determined by U_t as needed (since $\mathcal{D}$ is deterministic). For $0 \leq t < M$, there is always at most one index $j \in [k]$ for which $X_{j,t+1} \neq X_{j,t}$, and this index is determined by U_t (again since $\mathcal{D}$ is deterministic). Thus, conditioned on U_t, the variables $X_{1,t+1}, \ldots, X_{k,t+1}$ are independent. Using part 2 of Lemma 11 and the fact (Lemma 7) that $\boldsymbol{x}^j \sim \mu^{(u_t^j)}$ conditioned on U_t, we have $\mathbb{E}[X_{j,t+1}|U_t] \leq X_{j,t}$ for each j.

Now consider the final, added step. Condition on any value of $U_M = (u_M^1, \ldots, u_M^k)$. Lemma 7 tells us that $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$ 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 11. If $|u_M^j| > T$ then $\mathbb{E}[X_{j,M+1}] = |B|^{-1} = X_{j,M}$.

Thus the assumptions of Lemma 9 are satisfied, with $p_j = X_{j,0} \leq 1 - \varepsilon$. We conclude that for any monotone $\mathcal{C} \subseteq \mathcal{P}([k])$,

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{C}] \leq \Pr[D \in \mathcal{C}], \tag{3}$$

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

To prove statement 1 of the Theorem, let $\mathcal{C} := \mathcal{A}$. Note that $S[\boldsymbol{x}]$ and $u_M^1, \ldots, u_M^k$ are determined by $\boldsymbol{x}$, since $\mathcal{D}$ is deterministic. Condition on any value of $\boldsymbol{x}$ for which $S[\boldsymbol{x}] \in \mathcal{A}$. Under this conditioning, if $j \in [k]$ satisfies $|u_M^j| \leq T$ and $j \in S[\boldsymbol{x}]$, 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,

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

which in combination with (3) implies

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

as needed. To prove statement 2 of the Theorem, let $\mathcal{C} := N_{\alpha\varepsilon k}(\mathcal{A})$ in Eq. (3): we find

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]. \tag{4}$$

Arguing as above, $S[\boldsymbol{x}] \backslash \{j \in [k] : X_{j,N} = 1\}$ is always a set of size less than $\alpha\varepsilon k$, so $[S[\boldsymbol{x}] \in \mathcal{A}]$ implies $[\{j \in [k] : X_{j,N} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})]$. Thus, Eq. (4) implies $\Pr[S[\boldsymbol{x}] \in \mathcal{A}] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]$. □

Part 1 of Theorem 10 is a proper generalization of Theorem 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 immediate corollary, we obtain the following threshold DPT for relation problems, which specializes to an ordinary DPT for this setting (statement 3 in the Theorem below).

Theorem 12. *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 $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \sim \mu^{\otimes k}$, define the (random) set $S[\boldsymbol{x}]$ as in Theorem 10. Then if $\mathcal{R}$ is $\alpha\varepsilon Tk$-query-bounded for $\alpha \in (0, 1]$, we have:*

1. $\Pr[|S[\boldsymbol{x}]| \geq \eta k] \leq |B|^{\alpha\varepsilon k} \cdot \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq \eta k]$, *and also*

2. $\Pr[|S[\boldsymbol{x}]| \geq \eta k] \leq \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (\eta - \alpha\varepsilon)k]$.

3. $\Pr[|S[\boldsymbol{x}]| = [k]] \leq \min\{|B|^{\alpha\varepsilon k}(1-\varepsilon)^k, \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (1-\alpha\varepsilon)k]\}$. *If* $\alpha \leq 1/2$ *the second bound in the* min *is at most*

$$[1 - \varepsilon + 21\alpha \ln(1/\alpha)\varepsilon]^k .$$

Proof. Apply parts 1 and 2 of Theorem 10, with the choice $\mathcal{A} := \{A \subseteq [k] : |A| \geq \eta k\}$. We have $\Pr[D \in \mathcal{A}] = \Pr[D_1 + \ldots + D_k \geq \eta k]$, 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, $\Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})] = \Pr[D_1 + \ldots + D_k \geq (\eta - \alpha\varepsilon)k]$, 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 5 with $\beta := \alpha$, $\delta := \varepsilon$. □

Theorem 4 in the Introduction follows from the special case of Theorem 12 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, 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 12 will tend to give better results.

Note that part 2 of Theorem 12, 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 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 10 and 12, by the same reduction to the average-case setting used to prove Theorem 2.

6.3 Direct product theorems for learning tasks

Theorems 10 and 12 readily imply direct product theorems for the query complexity of learning tasks, as we explain next. Consider the scenario in a randomized algorithm $\mathcal{R}$ is given query access to some unknown function $h : \{0,1\}^n \to \{0,1\}$ drawn from some distribution μ over 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, such that $\mathbf{close}(h, \tilde{h})$ holds, where $\mathbf{close} \subseteq \mathcal{H} \times \mathcal{H}$ is some symmetric relation (assume $\mathbf{close}(h, h)$ always holds).

This task can be equivalently modeled as the relation problem associated with 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 don't 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 10 and 12 apply.

7 Proof of the XOR lemma

The proof of our XOR Lemma, Theorem 3 from the Introduction, is modeled on the proof of our threshold DPTs, and reuses Lemma 9.

Proof of Theorem 3. As usual we first set up some preliminaries. For a deterministic algorithm $\mathcal{D}$ over n input bits define

$$W_{\oplus}(u, \mathcal{D}) := 2 \cdot \Pr_{\boldsymbol{y} \sim \mu^{(u)}}[\mathcal{D}(\boldsymbol{y}) = f(\boldsymbol{y})] - 1.$$

If $|u| \leq T$, let $W^{\star}_{\oplus}(u) := \max_{\mathcal{D}} W_{\oplus}(u, \mathcal{D})$, where the maximum ranges over all deterministic algorithms making at most $T - |u|$ queries.

Lemma 13. *1.* $W^{\star}_{\oplus}(*^n) \leq 1 - 2\varepsilon$.

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

Lemma 13 follows immediately from Lemma 6, since $W^{\star}_{\oplus}(u) = 2W^{\star}(u) - 1$.

Now we prove the Theorem. As in the proof of Theorem 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 $f^{\oplus k}(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k)$ on inputs $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k \sim \mu^{\otimes k}$. Recall the definitions of u^j_t (for $j \in [k], 0 \leq t \leq M$) from Theorem 1. For a deterministic algorithm $\mathcal{D}$ define $\{X_{j,t}\}_{j \in [k], 0 \leq t \leq M}$ as follows: if $|u^j_t| \leq T$, set $X_{j,t} := W^{\star}_{\oplus}(u^t_j)$; otherwise, set $X_{j,t} := 0$.

We will extend the random sequences $\{X_{j,t}\}$ for one additional (non-query) 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^j_M, as follows. Let $b_j \in \{0,1\}$ be defined as the likeliest value of $f(\boldsymbol{y})$, where $\boldsymbol{y} \sim \mu^{(u^j_M)}$ (break ties arbitrarily). Let $r_j := \Pr[f(\boldsymbol{y}) = b_j] \in [1/2, 1]$, where again $\boldsymbol{y} \sim \mu^{(u^j_M)}$. Let $a_j := 2r_j - 1 \in [0,1]$.

If $|u^j_M| > T$, set $X_{j,M+1} := 0$. If instead $|u^j_M| \leq T$, our random process 'inspects' the actual value of the bit $f(\boldsymbol{x}^j)$ to help determine $X_{j,M+1}$. If $f(\boldsymbol{x}^j) \neq b_j$, let $X_{j,M+1} := 0$. If $f(\boldsymbol{x}^j) = b_j$, 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,

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

where the last inequality holds by the definition of $W^{\star}_{\oplus}(u^j_M)$ since $|u^j_M| \leq T$.

Let $\mathcal{U} = (U_0, \ldots, U_M)$, where $U_t := (u^1_t, \ldots, u^k_t)$. By an argument analogous to that in the proof of Theorem 10, we verify that $(\mathcal{X}, \mathcal{U})$ obey the assumptions of Lemma 9, this time with $p_j := 1 - 2\varepsilon$ (since $X_{j,0} \leq 1 - 2\varepsilon$). Applying Lemma 9 to $\mathcal{A} := \{A \subseteq [k] : |A| > (1 - \alpha\varepsilon)k\}$, we find

$$\Pr[|\{j : X_{j,M+1} = 1\}| > (1 - \alpha\varepsilon)k] \leq \Pr[D \in \mathcal{A}], \tag{5}$$

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 $F := [U_M = (u^1_M, \ldots, u^k_M), X_{1,M+1} = z_1, \ldots, X_{k,M+1} = z_k]$. Note that conditioning on F does *not* condition on the particular values $f(\boldsymbol{x}^j)$ which 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,

$$\begin{aligned}\Pr[f(\boldsymbol{x}^{j^\star}) = b_{j^\star}|F] &= \Pr[f(\boldsymbol{x}^{j^\star}) = b_{j^\star}|F' \wedge X_{j^\star,M+1} = 0] \\ &= \frac{\Pr[f(\boldsymbol{x}^{j^\star}) = b_{j^\star} \wedge X_{j^\star,M+1} = 0|F']}{\Pr[X_{j^\star,M+1} = 0|F']}\end{aligned}$$

$$= \frac{\Pr[f(\boldsymbol{x}^{j^\star}) = b_{j^\star}|F'] \cdot \Pr[X_{j^\star,M+1} = 0|F', f(\boldsymbol{x}^{j^\star}) = b_{j^\star}]}{\Pr[f(\boldsymbol{x}^{j^\star}) = b_{j^\star}|F'] \cdot \Pr[X_{j^\star,M+1} = 0|F', f(\boldsymbol{x}^{j^\star}) = b_{j^\star}] + \Pr[f(\boldsymbol{x}^{j^\star}) \neq b_{j^\star}|F'] \cdot \Pr[X_{j^\star,M+1} = 0|F', f(\boldsymbol{x}^{j^\star}) \neq b_{j^\star}]}$$

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

(using the fact that $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$ are independent conditioned on U_M, by Lemma 7, 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, $f(\boldsymbol{x}^{j^\star})$ is an unbiased random bit conditioned on F. Consequently, $f^{\oplus k}(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) = f(\boldsymbol{x}^{j^\star}) \oplus f^{\oplus k-1}(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{j^\star-1}, \boldsymbol{x}^{j^\star+1}, \ldots, \boldsymbol{x}^k)$ is an unbiased random bit conditioned on F. Thus under this conditioning, $\mathcal{D}$'s output bit equals the k-fold XOR with probability no more (and no less) than $1/2$. 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

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

using Eq. (5).

Finally, to get the concrete bound claimed in the Theorem statement, first suppose $\varepsilon = 1/2$; in this case the bound follows easily since $Y = 0$ with certainty. Now if $\varepsilon < 1/2$, note that $(1 - \alpha\varepsilon)k = (1 - (\alpha/2)(2\varepsilon))$, and apply Lemma 5 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. As a corollary we will obtain a direct product theorem for *errorless heuristics*, defined in Section 8.2.

8.1 Search problems

We need some preliminary definitios. 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 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 14. *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, 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 $W_V^\star(u) := \max_{\mathcal{D}}(W_V(u, \mathcal{D}))$, where the maximum ranges over all deterministic algorithms making at most $T - |u|$ queries. In other words, $W_V^\star(u)$ is the maximum success probability of any $(T-|u|)$-query algorithm in solving V on an input $\boldsymbol{y} \sim \mu^{(u)}$, where we reveal the bits described by u 'for free' to the algorithm. Then we have:

Lemma 15. *1. $W_V^\star(*^n) \leq 1-\varepsilon$.*

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

We omit the proof, which is essentially the same as that of Lemma 6.

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 $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k \sim \mu^{\otimes k}$. 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 $X_{j,t} := W_V^\star(u_t^j)$ if $|u_t^j| \leq T$, otherwise $X_{j,t} := 0$.

Unlike in Theorem 10, 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 10, we verify that the assumptions of Lemma 9 are satisfied, with $p_j = X_{j,0} \leq 1 - \varepsilon$ (Lemma 15, part 1).

Applying Lemma 9 to the monotone set $N_{\alpha\varepsilon k}(\mathcal{A})$, we conclude that

$$\Pr[\{j \in [k] : X_{j,M} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})], \tag{6}$$

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, there are two possibilities: 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. (6) yields the Theorem. □

From Theorem 14, we directly get an 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 16. *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 14 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 $[D_1 + \ldots + D_k > (\eta - \alpha\varepsilon)k]$, where $D_j := \mathbf{1}_{[j\in D]}$. These indicator variables are independent with expectation $1 - \varepsilon$. □

Theorem 17. *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 16 with $\eta := 1$. □

8.2 Errorless heuristics

An *errorless heuristic* for a (non-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 $\mathrm{Suc}^{\text{0-err}}_{T,\mu}(f) := 1 - \varepsilon$, 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 $\mathrm{Suc}^{\text{0-err}}_{T,\mu}(f)$ is exactly $\mathrm{Suc}_{T,\mu}(V_f)$, where the search problem V_f is defined as $V_f := \{u \in \{0,1,*\}^n : u$ 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 17:

Theorem 18. *Suppose* $\mathrm{Suc}^{0\text{-err}}_{T,\mu}(f) \leq 1-\varepsilon$. *Then for* $\alpha \in (0,1]$,

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

Note that an errorless heuristic to compute the k-fold XOR $f^{\oplus k}$ cannot produce any output other than '?', unless it has succeeded in determining $f^{\otimes k}$. Thus Theorem 18 also implies an XOR lemma with the same success bound for errorless heuristics.

Next we prove a worst-case analogue of Theorem 18. 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 [Yao77]; we include a proof for completeness.

Lemma 19. *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 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. Then $\mathcal{R}'(x) = f(x)$ on every input. Also, the expected number of queries of $\mathcal{R}'$ on any input is strictly less than

$$\begin{aligned}\sum_{m\geq 1}(1-\eta)^{m-1}\eta\,(m\cdot\eta R_0(f)) &= (\sum_{m\geq 1}(1-\eta)^{m-1}m)\cdot\eta^2 R_0(f)\\ &= \frac{1}{\eta^2}\cdot\eta^2 R_0(f)\\ &= R_0(f),\end{aligned}$$

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

Theorem 20. *For* $\alpha \in (0,1/2]$, *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* $[22\alpha\ln(1/\alpha)]^k$.

Proof. Set $\gamma := \alpha/2$. Let μ_γ be the distribution given by Lemma 19, so that $\mathrm{Suc}^{0\text{-err}}_{\gamma R_0(f),\mu_\gamma}(f) \leq \gamma$. By Theorem 18 applied to α, with $T := \gamma R_0(f)$ and $\varepsilon := 1-\gamma$,

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

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

$$\begin{aligned}
\mathrm{Suc}^{\text{0-err}}_{\alpha^2 R_0(f)k/4, \mu_\gamma^{\otimes k}}(f^{\otimes k}) &\leq \Pr_{Y \sim B_{k,\gamma}}[Y > (1-\alpha(1-\gamma))k] \\
&< [1-(1-\gamma)+21\alpha\ln(1/\alpha)(1-\gamma))]^k \\
&\text{(applying Lemma 5, with } \beta := \alpha \leq 1/2 \text{ and } \delta := (1-\gamma)) \\
&< [\alpha/2 + 21\alpha\ln(1/\alpha)]^k \\
&< [22\alpha\ln(1/\alpha)]^k .
\end{aligned}$$

□

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 21. *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. Also note that, by convexity, Theorem 21 also bounds the success probability of any 'randomized size-$T^{\alpha\varepsilon k}$ algorithm' $\mathcal{R}$, i.e., of any probability distribution over size-$T^{\alpha\varepsilon k}$ decision trees.

Proof. The proof follows that of Theorem 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. As in Theorem 1, let $W(u,\mathcal{D}) := \Pr_{\boldsymbol{y}\sim\mu^{(u)}}[\mathcal{D}(\boldsymbol{y}) = f(\boldsymbol{y})]$. Given a real number $Z \in [1,T]$, let $W^\star_{\mathrm{size}}(u,Z) := \max_{\mathcal{D}} W(u,\mathcal{D})$, where the maximum is over all decision trees $\mathcal{D}$ of size at most Z.

Lemma 22. *1.* $W^\star_{\mathrm{size}}(*^n, T) \leq 1-\varepsilon$.

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

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

The proof is very similar to that of Lemma 6, 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 $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \sim \mu^{\otimes k}$. 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 $\boldsymbol{x}^j$ after $\mathcal{D}$ has taken t steps (where 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 (so, $S_t = 1$ iff $\mathcal{D}$ 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 $\boldsymbol{x}^j$, we again let $Z_{j,t+1} := Z_{j,t}$. If the $(t+1)$-st query is to $\boldsymbol{x}^j$, let $Z_{j,t+1} := \frac{S_{t+1}}{S_t} \cdot Z_{j,t}$.

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

Condition on any complete execution of $\mathcal{D}$, as described by $u^1_M, \ldots, u^k_M$. 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 $f(\boldsymbol{x}^j)$ 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 $f^{\otimes k}(\boldsymbol{x})$ is at most $2^{|B|}P_M$, 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,

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

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 dyamic, 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.

To make our scenario concrete, we first define the type of entity with which our query algorithms interact. Define an *interactive automaton (IA)* as a 5-tuple

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

- $\mathbf{seeds}, \mathbf{states}, \mathbf{queries}$ are each finite sets, and $\mathbf{states}$ 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.*

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 $(z, \Delta(z, s, q))$.

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 state and prove 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 $\xi(z) \in \{0,1\}^{m+(|\,\mathbf{queries}\,|+1)^N}$. 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, r-th 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 r-th step are $q_1, \ldots, q_r$, then for its r-th query to $\xi(z)$, $\mathcal{R}$ looks at the entry $(q_1, \ldots, q_r, *, *, \ldots, *)$ to learn $\mathcal{M}$'s r-th response. Call an algorithm 'interaction-faithful' if its sequence of queries to any input string always obey this format.

Obviously, 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 that of 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 easy 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's 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 $u \in \{0,1,*\}^n$, $\mathcal{R}$ 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 queries which 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 k 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 j-th 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 10) for standard query algorithms obeying a restriction map.

Theorem 23. *Let $P \subseteq \{0,1\}^n \times B$ be a total relation such that any T-query, τ-faithful algorithm solves the relation problem for P with probability at most $1 - \varepsilon$ under input distribution μ.*

For any algorithm $\mathcal{R}$ making queries to inputs $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \sim \mu^{\otimes k}$ and producing output in B^k, define the random set $S[\boldsymbol{x}]$ as in Theorem 10.

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 10 also hold for the present setting.

Proof. (Sketch) The proof follows that of Theorem 10; we only describe the differences. For $u \in \{0,1,*\}^n$, and for a deterministic algorithm $\mathcal{D}$ on n input bits, let $W_P(u, \mathcal{D}) := \Pr_{\boldsymbol{y} \sim \mu^{(u)}}[(\boldsymbol{y}, \mathcal{D}(\boldsymbol{y})) \in P]$ as in Theorem 10. Also, say that $\mathcal{D}$ is *u-inducing* if, on any input $x \in \{0,1\}^n$ which extends[6] u, the outcome of $\mathcal{D}$'s first $|u|$ queries to x are described by u.

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

Lemma 24. *1. $W^\star_{P,\tau}(*^n) \leq 1 - \varepsilon$.*

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

The rest of the proof follows Theorem 10, with $W^\star_{P,\tau}(u)$ taking the place of $W^\star_P(u)$. □

One can also prove a DPT for search problems for τ-faithful query algorithms, along the lines of Theorem 14. 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

- Can the bounds in our threshold DPTs and XOR lemma be improved? For the threshold DPTs, how does the tightness of these bounds depend on the monotone set $\mathcal{A}$?

[6] (as defined in Section 8.1)

- 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 $\alpha \gg 1$. (As mentioned earlier, $O(R_2(f)k \log k)$ queries always suffice to achieve high success probability.) The corresponding question in the quantum query model was settled by Buhrman et al. [BNRdW05].

  Our Theorem 1 would help resolve this question, if we could identify a function f and distribution μ for which $\mathrm{Suc}_{\alpha R_2(f),\mu}(f)$ approaches 1 not-too-quickly as α grows (say, not faster than $1 - 2^{-O(\alpha)}$, for α in a reasonable range). The AND/OR-tree evaluation problem, whose randomized query complexity was studied in [SW86], might be a good candidate.

- Can our direct product theorem be extended to the quantum query model? The main difficulty is that the natural analogue of the conditional-independence property we used in Theorem 1 is false for quantum query algorithms. The close connection shown by Reichardt [Rei09] between quantum query complexity and *span programs* may be useful in pursuing this question.

- Can the ideas in this paper help improve our understanding of the direct product problem in the communication and circuit models, or other computational settings?

12 Acknowledgements

I thank Ronald de Wolf for numerous helpful comments, and in particular for encouraging me to look at threshold DPTs.

References

[ASdW09] Andris Ambainis, Robert Spalek, and Ronald de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. *Algorithmica*, 55(3):422–461, 2009.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[BNRdW05] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. In *STACS*, pages 593–604, 2005.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition, 2009.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(50), 1995.

[Hol05] Thomas Holenstein. Key agreement from weak bit agreement. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 664–673. ACM, 2005.

[HS10] Thomas Holenstein and Grant Schoenebeck. General hardness amplification of predicates and puzzles. *CoRR*, abs/1002.3534, 2010.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[IK10] Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 17(072), 2010.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *FOCS*, pages 538–545, 1995.

[IRW94] Russell Impagliazzo, Ran Raz, and Avi Wigderson. A direct product theorem. In *Structure in Complexity Theory Conference*, pages 88–96, 1994.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

[JKS10] Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity, 2010, arXiv:1004.0105.

[Kla09] Hartmut Klauck. A strong direct product theorem for disjointness. *CoRR*, abs/0908.2940, 2009.

[KŠdW07] Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM J. Comput.*, 36(5):1472–1493, 2007.

[LSŠ08] Troy Lee, Adi Shraibman, and Robert Špalek. A direct product theorem for discrepancy. In *IEEE Conference on Computational Complexity*, pages 71–80, 2008.

[Nis89] Noam Nisan. Crew prams and decision trees. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 327–335, New York, NY, USA, 1989. ACM.

[NRS99] Noam Nisan, Steven Rudich, and Michael E. Saks. Products and help bits in decision trees. *SIAM J. Comput.*, 28(3):1035–1050, 1999.

[Rei09] Ben Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *FOCS*, pages 544–551. IEEE Computer Society, 2009.

[Sha03] Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1-2):1–22, 2003.

[SW86] Michael E. Saks and Avi Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *FOCS*, pages 29–38, 1986.

[Ung09] Falk Unger. A probabilistic inequality with applications to threshold direct-product theorems. In *FOCS*, pages 221–229, 2009.

[Š08] Robert Špalek. The multiplicative quantum adversary. In *IEEE Conference on Computational Complexity*, pages 237–248, 2008.

[VW08] Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4(1):137–168, 2008.

[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *FOCS*, pages 222–227, 1977.

A Proof of Lemma 5

We use a general form of Chernoff's inequality:

Lemma 25 ([DP09], §1.3)**.** *Suppose $Y \sim B_{k,p}$, with $q := 1 - p$. Then for $t \in [0, q)$,*

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

To prove Lemma 5, apply Lemma 25 with $t := (1-\beta)\delta$. We have

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

Now $\beta\delta \leq 1/2$ implies $(1-\beta\delta)^{-1} \leq (1+2\beta\delta)$. Also,

$$\beta^{-\beta\delta} = e^{\beta \ln(1/\beta)\delta} \leq \left(e^{2\beta \ln(1/\beta)\delta} \right)^{1-\beta\delta},$$

since $1 - \beta\delta \geq 1/2$. By our remarks on the function $x \ln(1/x)$ in Section 2.2, we have $2\beta \ln(1/\beta)\delta < 2e^{-1}\delta < 1$. Now we claim that $e^x \leq 1 + (e-1)x \leq 1 + 1.8x$ for $x \in [0,1]$. To see this, just note that e^x is convex on $\mathbb{R}$, that $e^0 = 1$, and that $e^1 = 1 + (e-1) \cdot 1$. Thus, $\left(e^{2\beta \ln(1/\beta)\delta}\right)^{1-\beta\delta} \leq (1 + 1.8 \cdot 2\beta \ln(1/\beta)\delta)^{1-\beta\delta}$. Plugging in these observations,

$$\begin{aligned} \Pr[Y > (1-\beta\delta)k] &\leq [(1-\delta)(1+2\beta\delta)(1+3.6\beta \ln(1/\beta)\delta)]^{(1-\beta\delta)k} \\ &< [(1-\delta)(1+3\beta \ln(1/\beta)\delta)(1+3.6\beta \ln(1/\beta)\delta)]^{(1-\beta\delta)k} \\ &< [(1-\delta)(1+11\beta \ln(1/\beta)\delta)]^{(1-\beta\delta)k} \\ &< [1-\delta+11\beta \ln(1/\beta)\delta]^{(1-\beta\delta)k} \\ &= \frac{[1-\delta+11\beta \ln(1/\beta)\delta]^k}{[1-\delta+11\beta \ln(1/\beta)\delta]^{\beta\delta k}}. \end{aligned} \tag{7}$$

We bound the denominator in Eq. (7) by splitting into cases. First assume $\delta \leq 1/2$. In this case we bound

$$\begin{aligned}[1-\delta+11\beta\ln(1/\beta)\delta]^{\beta\delta k} &\geq (1-\delta)^{\beta\delta k} \\ &\geq 2^{-\beta\delta k} = e^{-(\ln 2)\beta\delta k}.\end{aligned}$$

Next assume $\delta > 1/2$. In this case we bound

$$\begin{aligned}[1-\delta+11\beta\ln(1/\beta)\delta]^{\beta\delta k} &> (7\beta\delta)^{\beta\delta k} \\ &= e^{-\beta\delta\ln(1/(7\beta\delta))k} > e^{-\beta\delta\ln(1/\beta)k}.\end{aligned}$$

In either case, we conclude that Eq. (7) is less than

$$\begin{aligned}\left[(1-\delta+11\beta\ln(1/\beta)\delta)\, e^{\beta\delta\ln(1/\beta)}\right]^k &\leq \left[(1-\delta+11\beta\ln(1/\beta)\delta)\,(1+1.8\beta\ln(1/\beta)\delta)\right]^k \\ &< \left[1-\delta+21\beta\ln(1/\beta)\delta\right]^k.\end{aligned}$$

B Proof of Lemma 7

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 j-th 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 $[\boldsymbol{x}^{j'} = x^{j'}\ \forall j' \neq j]$, the j-th input $\boldsymbol{x}^j$ is distributed according to $\mu^{(u_t^j)}$. This will prove the Lemma.

Consider each $y \in \{0,1\}^n$ which 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 $\boldsymbol{x}^j = y$ is

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

by definition of $\mu^{(u_t^j)}$. This completes the proof.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de