Products and Help Bits in Decision Trees

Preliminary Version

Noam Nisan [*] Steven Rudich [†] Michael Saks [‡]

October 18, 1994

Abstract

We investigate two problems concerning the complexity of evaluating a function f at a k-tuple of unrelated inputs by k parallel decision tree algorithms.

In *the product problem*, for some fixed depth bound d, we seek to maximize the fraction of input k-tuples for which all k decision trees are correct. Assume that for a single input to f, the best decision tree algorithm of depth d is correct on a fraction p of inputs. We prove that the maximum fraction of k-tuples on which k depth d algorithms are all correct is at most p^k, which is the trivial lower bound. We show that if we replace the depth d restriction by "expected depth d", then this result fails.

In the help-bit problem, we are permitted to ask $k-1$ arbitrary binary questions about the k-tuple of inputs. For each possible $k-1$-tuple of answers to these queries we will have a k-tuple of decision trees which are supposed to correctly compute all functions on k-tuples that are consistent with the particular answers. The complexity here is the maximum depth of any of the trees in the algorithm. We show that for all k sufficiently large, this complexity is equal to $deg^s(f)$ which is the minimum degree of a multivariate polynomial whose sign is equal to f.

Finally, we give a brief discussion of these problems in the context of other complexity models.

[*] Computer Science Department, Hebrew University, Jerusalem, Israel. Supported by BSF grant 92-00043 and by a Wolfson award administered by the Israeli Academy of Sciences.

[†] Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa. Partially supported by NSF grant CCR-9119319.

[‡] Department of Mathematics, Rutgers University, New Brunswick, NJ 08903. Supported in part by NSF contracts CCR-9215293 and STC–91–19999 and by DIMACS

1 Introduction

Pick your favorite computation model and complexity measure, e.g. boolean circuit size, communication complexity, decision tree depth, interactive proof length, tensor rank, etc. Any attempt to understand such a model and complexity measure requires understanding the ways that an "unreasonable" computation can be more efficient than a "reasonable" one. Of course, what is "reasonable" changes as our understanding of the model improves.

Suppose we are given several unrelated instances of a problem to solve. The "reasonable" approach is to solve each instance separately; intuitively, any computation that is useful for solving one instance is irrelevant to any of the others. To what extent is this intuition valid in a given model? The following question is the most common way of formalizing this.

The Direct-sum problem: Suppose that the complexity of computing some function f is c. Is it true that computing f twice, on two unrelated inputs requires complexity $2c$? How about computing f on k unrelated inputs?

This question was introduced (to our knowledge) in the context of bilinear circuits [J, Bsh], and has been studied also in the contexts of boolean circuits [FKN] and communication complexity [KRW]. In this paper we consider two related problems of a similar flavor:

The Product Problem: Let f be a function and suppose that for any complexity c computation, the fraction of inputs on which it correctly computes f is at most p. Suppose that we have two independent computations, each taking as input an ordered pair a, b of inputs to f, where the first computation is trying to compute $f(a)$ and the second is trying to compute $f(b)$. If each of the two computations has complexity at most c, can the fraction of input pairs a, b on which both are correct exceed p^2? What about the analogous question for k independent computations and k inputs?

If the first computation only uses the input a and the second only uses the input b, then the p^2 upper bound is trivial. Intuition suggests that there is no advantage in having each computation access the others input. A variant of this problem, in which we seek to compute f on the two inputs by a single computation was studied recently in [IRW].

The Help-bit Problem: Suppose that the complexity of computing the boolean function f is c. Suppose that one wishes to compute f on two inputs a and b, and is allowed for free one "help-bit", i.e. an arbitrary function of the two inputs. Is it possible to choose this help-bit function so that, given the help-bit, $f(a)$ and $f(b)$ can each be computed by a computation of complexity less than c, and if so, by how much? How about computing f on k inputs with $k-1$ help bits?

The help-bit problem was introduced (to our knowledge) in the context of constant depth circuits in [Cai], and was also studied in the context of boolean circuits in [ABG]. The point here is that if we have k inputs, then with k help bits we can use them to obtain the value of f on each of the inputs, and no further computation is necessary. With only $k-1$ help bits, we can for instance obtain the value of f at $k-1$ inputs, but then we still need complexity c to compute f on the last input. Is there a more effective use of the help bits?

In this paper we consider these problems in the context of the boolean decision tree complexity – perhaps the simplest computational model. The cost of a computation (decision tree) is simply the number of input variables that are read (the depth of the decision tree); a more precise definition is given in Section 2. While it is an easy exercise to see that "direct-sum" holds for decision tree depth, the other two problems are more difficult. Our answer for the product problem is a qualified "Yes":

Theorem 1 *Let f be an n-variable boolean function and suppose that any depth d decision tree computes f correctly on a fraction at most p of the inputs. Let $T_1, T_2, \ldots, T_k$ be decision trees that each access a set of nk variables corresponding to a k-tuple $a_1, a_2, \ldots, a_k$ of inputs to f. If each of the T_i have depth at most d, then the fraction of k-tuples $a_1, a_2, \ldots, a_k$ on which each T_i correctly outputs $f(a_i)$ is at most p^k.*

The theorem seems completely obvious; however, the reader might test her intuition on the following variation. Suppose that, in the above Theorem we change the complexity measure from "depth" to "average depth" , i.e, the average over all inputs of the depth of the leaf reached by the input. This modified statement of the Theorem seems similarly obvious, but, as we will see, it is false.

The recent work of [IRW], which was done independently of ours, includes a (substantially different) proof of a weaker variant of this theorem, namely that a *single* depth d tree that tries to compute all k functions can be correct on at most a p^k fraction of the inputs. Our result shows that even if we use k parallel decision trees then we can't do better than this.

For the help bit problem, the answer is more complicated. Nathan Linial [Lin] has shown that the complexity of computing f on two inputs with one help bit is at least $deg(f)$, the degree of the (unique) multilinear real polynomial that is equal to f. Since almost all boolean functions on n-variables have $deg(f) = n$, this says that help bits don't help for most functions. This result does not seem to extend to $k \geq 3$. In fact, for sufficiently large k our results imply that it is false. We manage to prove a lower bound that holds for all k, and is always tight when k, the number of instances to be solved, is sufficiently large. We need the following definitions. If f is an n-variate boolean function, we say that the n-variate real polynomial p *sign-represents* f if for all inputs a, $f(a) = sgn(p(a))$ where $sgn(z) = 1$ if $z > 0$ and $sgn(z) = -1$ otherwise (here we are taking our Boolean set to be $\{-1, 1\}$). The *sign-degree* of f, $deg^s(f)$, is the minimum degree of a polynomial that sign represents f.

Theorem 2 *Let f be an n-variate boolean function, and suppose that the optimal decision tree that computes f has depth d. Then for all $k \geq 1$, any solution to the help bit problem for f for k inputs and $k-1$ help bits requires depth at least $deg^s(f)$. Furthermore, for all sufficiently large k, there is a decision tree algorithm with $k-1$ help bits whose depth is $deg^s(f)$.*

In the case that f is equal to the product of n variables (which corresponds to the parity function for $\{0, 1\}$-valued variables), $deg^s(f) = n$ and so, the lower bound implies that help-bits don't help in this case. Actually, this function and its negative are the only functions with $deg^s(f) = n$. Since the ordinary decision tree complexity of most boolean functions is n, this means that for large enough k, the complexity of k instances given $k-1$ help bits is less than the ordinary decision tree complexity for most functions. In particular, if f is the majority function, then $deg^s(f) = 1$, and the lower bound is vacuous, while the upper bound says that for k sufficiently large, it is possible to ask $k-1$ binary questions so that, given the answers, the value of the function on any one of the k inputs can be computed by probing just one variable.

In the next section, we review the decision tree model. In Section 3 we give a general formulation for the product problem in decision trees, and prove a generalization (Theorem 3.1) of Theorem 1. In Section 4, we discuss the help bits problem and prove Theorem 2. Most proofs are in the appendices.

While some of the techniques we develop apply only to the decsion tree model, some of them may be applied to other models as well, and in fact suffice for obtaining many of the known results in the boolean circuit model. We sketch these applications in the last section.

2 Preliminaries

In this section we present some basic definitions and notation. Most of the notions discussed here are very familiar, but in some cases our notation is non-standard.

2.1 Boolean functions

For purposes of this paper it will be convenient to use $B = \{-1, 1\}$ as our Boolean domain, instead of $\{0, 1\}$. If X is a set, a *boolean assignment* to X is a map α from X to B. The set of boolean assignments to X is denoted B^X. We refer to the elements of X as *variables*. We will consider probability distributions over the set of assignments. For a specified distribution D, a random assignment chosen according to D is denoted by placing a $\tilde{}$ above the identifier, e.g., $\tilde{\alpha}$. A *boolean function* over the variable set X and range R, or (X, R)*-function* is a function from B^X to R. In this paper, the range R is always equal to B^k for some integer k.

2.2 Decision Trees

All trees in this paper are rooted, ordered, binary trees. For such a tree T every internal node v has exactly two children, and the two children are distinguished as the (-1)-child and (+1)-child of v. The depth $d_T(v)$ of a node v is, as usual, the number of edges along the path from v to the root and the depth d_T of T is the maximum depth of any node in T.

Formally, a *decision tree over the variable set X with range R* or (X, R)-decision tree is a triple (T, p, a) where T is a rooted binary tree, p is a map that associates to each internal node v a variable $x = p_v$ in the set X, and a is a map that associates each leaf v to an element a_v of R. The label p_v is called the *query* associated to v, and node v is said to *probe* variable p_v. We will generally say that T is an (X, R)-decision tree, keeping the maps p and a implicit. The set of (X, R)-decision trees over X is denoted $\mathcal{T}(X, R)$, or simply $\mathcal{T}$.

Let T be an (X, R)-decision tree. If α is any assignment in B^X, the *computation* of T on α, is the unique path $v^0, v^1, v^2, \ldots, v^s$ from the root of T to some leaf $v^s = l_T(\alpha)$ as follows: start from the root v^0 and inductively define define v^{i+1} for $i \geq 0$ as the $\alpha(p_{v^i})$-child of v_i. The output of the computation is the label $a_{l_T(\alpha)}$. Thus T can be viewed as a boolean function over X with range R. Trivially, every (X, R)-function f is computed by some (X, R)-decision tree.

The usual cost function for the computation performed by T on α is the length (number of internal nodes) of the computation path, denoted $C(T, \alpha)$. The worst case complexity $C(T)$ is the maximum over α of $C(T, \alpha)$. $C(f)$, the decision tree depth of f, is the minimum of $C(T)$ over all decision trees that compute f. For a distribution D on assignments, the distributional complexity $C_D(T)$ is the average of $C(T, \tilde{\alpha})$ with respect to the distribution D.

For a given (X, R)- function f, and a complexity bound b (with respect to some complexity measure), we are interested in how well f can be approximated by a tree of complexity at most b. The closeness of approximation is defined with respect to a probability distribution D on boolean assignments to X. Thus for each (X, R)-decision tree T, the *agreement probability* $q_D(f; T)$ of T with f relative to D, is the probability that $T(\tilde{\alpha}) = f(\tilde{\alpha})$, with respect to the random assignment $\tilde{\alpha}$ chosen according to D. The *decision tree approximation problem* for $(f, D, \mathcal{U})$ where f is an (X, R)-function, D is a distribution over boolean assignments to X, and $\mathcal{U}$ is a set of decision trees is to determine $q_D(f; \mathcal{U})$, which is defined to be the maximum agreement probability $q_D(f; T)$ over all $T \in \mathcal{U}$. Of particular interest is the case that $\mathcal{U}$ is the set $\mathcal{T}_d(X, R)$ of decision trees of depth at most d.

Finally, a *decision forest* F over X and ranges $R_1, R_2, \ldots, R_k$ is an ordered sequence $T_1, T_2, \ldots, T_k$, where T_i is an (X, R_i)-decision tree. F computes a boolean function from B^X to $R = R_1 \times R_2 \times \ldots \times R_k$.

3 The Product Problem

Let $X_1, X_2, \ldots, X_k$ be pairwise disjoint sets of variables, and let $D_1, D_2, \ldots, D_k$ be, respectively, distributions over assignments to $X_1, X_2, \ldots, X_k$. Let $X = X_1 \cup X_2 \cup \ldots \cup X_k$. A boolean assignment β for X will be viewed as a k-tuple $(\beta_1, \beta_2, \ldots, \beta_k)$ where β_i is an assignment for X_i. Let D denote the distribution over assignments to X given by $\mathbf{Prob}_D(\tilde{\alpha} = \beta) = \prod_{i=1}^k \mathbf{Prob}_{D_i}(\tilde{\alpha}_i = \beta_i)$, i.e., the product distribution $D_1 \times D_2 \times \ldots \times D_k$.

Now suppose that we have k decision tree approximation problems $(f_1, D_1, \mathcal{U}_1), (f_2, D_2, \mathcal{U}_2), \ldots, (f_k, D_k, \mathcal{U}_k)$, where for each i, f_i is a (X_i, R_i)-function, and let $q_i = q_{D_i}(f_i; \mathcal{U}_i)$ be the optimal agreement probability for $\mathcal{U}_i$ with f_i relative to D_i. It will be convenient sometimes to view f_i as a function of the entire variable set X that ignores all variables except those in X_i. We consider the problem of *simultaneously approximating* $f_1, f_2, \ldots, f_k$ by a decision forest $F = (T_1, T_2, \ldots, T_k)$ where

$T_i \in \mathcal{U}_i$. The *simultaneous agreement probability* $q_D(f_1, f_2, \ldots, f_k; T_1, T_2, \ldots, T_k)$ for $T_1, T_2, \ldots, T_k$ with $f_1, f_2, \ldots, f_k$ denotes the probability, for $\tilde{\alpha}$ chosen according to D, that $(T_1(\tilde{\alpha}) = f_1(\tilde{\alpha})) \wedge (T_2(\tilde{\alpha}) = f_2(\tilde{\alpha})) \wedge \ldots \wedge (T_k(\tilde{\alpha}) = f_k(\tilde{\alpha}))$. For $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k$ where $\mathcal{U}_i$ is a family of (X, R_i)-trees, $q_D(f_1, f_2, \ldots, f_k; \mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k)$ denotes the maximum of $q_D(f_1, f_2, \ldots, f_k; T_1, T_2, \ldots, T_k)$ over all choices of trees with $T_i \in \mathcal{U}_i$.

Now, since f_i only depends on X_i and D chooses the assignments $\tilde{\alpha}_1$, $\tilde{\alpha}_2$, $\ldots \tilde{\alpha}_k$ to $X_1, X_2, \ldots, X_k$ independently, it would seem that $q_D(f_1, f_2, \ldots, f_k; T_1, T_2, \ldots, T_k)$ should just be the product of the probabilties $q_{D_i}(f_i; T_i)$. This is clearly the case if each tree T_i only queries variables in X_i. However (as shown by the examples in below), if T_i is allowed to query variables outside of X_i, then this need not be the case. Intuitively, it would seem that variables outside of X_i could not help to approximate f_i and indeed this is trivially true, if we are only trying to approximate f_i. But when we seek to approximate all of the functions simultaneously, it is no longer obvious that such "cross-queries" are irrelevant.

Nevertheless, one might expect that for "reasonable" classes $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k$ of decision trees, the optimal simultaneous agreement probability is attained by a sequence of trees $T_1, T_2, \ldots, T_k$ with T_i querying variables only in X_i, and is thus equal to the product of the individual optimal agreement probabilities. The main result of this section is to prove this in the case that for each i, $\mathcal{U}_i$ is the set of trees of some fixed depth d_i.

Theorem 3.1 *Let $f_1, f_2, \ldots, f_k$ and $D_1, D_2, \ldots, D_k, D$ be as above. Let $d_1, d_2, \ldots, d_k$ be nonnegative integers. Then*

$$q_D(f_1, f_2, \ldots, f_k; \mathcal{T}_{d_1}, \mathcal{T}_{d_2}, \ldots, \mathcal{T}_{d_k}) = \prod_{i=1}^{k} q_{D_i}(f_i, \mathcal{T}_{d_i})$$

Note that Theorem 1 is a special case of the above. Before giving the proof we present two examples to show that multiplicativity fails for some natural alternative choices of the classes $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k$.

Example 3.1 *Theorem 3.1 fails if we replace the class $\mathcal{T}_{d_i}$ by the class $\mathcal{S}^i_{d_i}$ of trees that are restricted to query at most d_i variables from X_i along any path, but can query variables outside X_i for free. Consider the following trivial example. Let $k = 2$ and let $X_1 = \{x_1\}, X_2 = \{x_2\}$. The distribution D_1 assigns x_1 to 1 with probability 1/2, and D_2 assigns x_2 to 1 with probability 1/2. The functions f_1 and f_2 are given by $f_1(x_1) = x_1, f_2(x_2) = x_2$. Now let $d_1 = d_2 = 0$. This means that we do not allow T_1 to look at any variables in X_1 and we do not allow T_2 to look at any variable in X_2. Clearly $q_{D_1}(f_1, \mathcal{S}^1_0) = q_{D_2}(f_2, \mathcal{S}^2, 0) = 1/2$. However, we can achieve simultaneous* *agreement probability better than 1/4. Let T_1 be the tree that queries x_2 and outputs x_2 and T_2 be the tree that queries x_1 and outputs x_1. Then, the probability that both T_1 and f_1 agree and T_2 and f_2 agree is just the probability that x_1 and x_2 are assigned the same value, which is 1/2.*

A somewhat more subtle example is given by:

Example 3.2 *For a distribution D over B^X, let $\mathcal{T}_d^D$ be the class of trees whose expected depth with respect to D is d, i.e., $T \in \mathcal{T}_d$ if the average number of variables queried, with respect to $\tilde{\alpha}$ chosen from D is at most d. Then the above theorem is false if we replace $\mathcal{T}_{d_i}$ by $\mathcal{T}_{d_i}^{D_i}$. To see this, let X be a set of three variables, and f be the parity function on f. Let U be the uniform distribution over assignments to X and let $d = 2$. First we show that the maximum agreement probability with f attained by a decision tree S of expected depth at most 2, is 3/4. Agreement probability 3/4 is attained by the tree S that queries a particular variable x, and if it is 0, then it returns 0, and otherwise it queries the remaining two variables and returns the parity of them.*

Now let X_1, f_1, D_1 and X_2, f_2, D_2 be copies of X, f, U on disjoint variable sets, and let S_1 and S_2 denote the corresponding copies of the decision tree S described above. We show that it is possible to choose decision trees T_1, T_2 each of expected depth at most 2, whose agreement probability exceeds $9/16 = (3/4)^2$. Let $T_1 = S_1$, and let x_1 denote the variable in X_1 probed first by T_1. Let T_2 be the following tree: first probe x_1 (in X_1). If it is 0, output 0. If it is one, then perform tree S_2. Clearly, the expected depth of this tree is 2. Now, let us consider the probability of the event A that both $T_1(\tilde{\alpha}) = f_1(\tilde{\alpha})$ and $T_2(\tilde{\alpha}) = T_2(\tilde{\alpha})$. Then $\mathbf{Prob}_D(A) = 1/2(\mathbf{Prob}_D(A|x_1 = 0) + \mathbf{Prob}(A|x_1 = 1)$. The conditional probability of A given $x_1 = 0$ is 1/4. If $x_1 = 1$ then T_1 must agree with f_1, while T_2 will agree with f_2 with probability $3/4$. Thus the probability of simultaneous agreement is $5/8$.

What happens in the above example is that the variable x_1 acts as a shared random coin that partially coordinates the two computations so that they are more likely to be simultaneously correct.

Proof of Theorem 3.1 Fix a sequence $T_1, T_2, \ldots, T_k$ of decision trees with T_i of depth at most d_i. For $I \subseteq [k] = \{1, 2, \ldots, k\}$, let $C(I)$ denote the event $\wedge_{i \in I}(T_i = f_i(x^i)$, i.e., the event that all of the trees indexed by I evaluate their respective functions correctly. We seek to prove that $\mathbf{Prob}[C([k])]$ is bounded above by $\prod_{i=1}^{k} q_{D_i}(f_i, \mathcal{T}_{d_i})$.

The proof is by induction on k, and for fixed k by induction on the k-tuple

$d_1 + d_2 + \ldots + d_k$. The result is vacuous if $k = 1$.

So assume that $k \geq 2$. Consider first the case that $d_i = 0$ for some i. We may assume that $d_k = 0$. Thus, the k^{th} party must guess the value of $f_k(\tilde{\alpha}_k)$ without looking at any variables, so T_k consists of a single leaf labeled -1 or 1. Now, by conditioning on the value of the vector $\tilde{\alpha}_k$, the probability, P^* that $C([k])$ holds can be written:

$$
\begin{aligned}
P^* &\leq \sum_{\beta \in B^{X_k}} \mathbf{Prob}[\tilde{\alpha}_k = \beta] \times \mathbf{Prob}[T_k(\beta) = f_k(\beta)] \times \mathbf{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \\
&\leq \max_{\beta \in B^{X_k}} \mathbf{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \times \sum_{\beta \in B^n} \mathbf{Prob}[\tilde{\alpha}_k = \beta] \mathbf{Prob}[T_k(\beta) = f_k(\beta)] \\
&= \max_{\beta \in B^{X_k}} \mathbf{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \times \mathbf{Prob}[T_k(\tilde{\alpha}) = f_k(\tilde{\alpha}))] \\
&\leq \max_{\beta \in B^{X_k}} \mathbf{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \times q_{D_k}(f_k, 0).
\end{aligned}
$$

Now let γ be the assignment of $\tilde{\alpha}_k$ that maximizes the probability in the last expression. For each i between 1 and $k-1$, define the tree U_i by contracting T_i using $\tilde{\alpha} = \gamma$. Then we may rewrite the last term as

$$
\mathbf{Prob}[(U_1(\tilde{\alpha}) = f_1(\tilde{\alpha})) \wedge \ldots (U_{k-1}(\tilde{\alpha}) = f_{k-1}(\tilde{\alpha}))] \times q_{D_k}(f_k, 0)
$$

Each tree U_i has depth at most d_i, and so we may bound the first factor by $q_D(f_1, f_2, \ldots, f_{k-1}; \mathcal{T}_{d_1}, \mathcal{T}_{d_2}, \ldots, \mathcal{T}_{d_{k-1}})$ which by the induction hypothesis equals $\prod_{i=1}^k q_{D_i}(f_i, \mathcal{T}_{d_i})$. Thus the desired result follows.

Now we assume that $d_i > 0$ for all i. Define a directed graph on $\{1, 2, ..., k\}$ with an edge from i to j if the first variable probed by T_i is an input to f_j. Since this directed graph has outdegree one, it has a directed cycle. Let $j \geq 1$ be the length of the cycle. Let us rename the set of indices in the cycle by the set $[j] = \{1, 2, \ldots, j\}$ in such a way that for each $i < j$, the first probe of T_i is a variable, denoted x_{i+1}, in X_{i+1} and the first probe of T_j is a variable, denoted x_i, in X_1.

The intuition behind the rest of the proof is that for $i \in [j]$, it is possible to replace each tree T_i by trees of the same depth in which the first probe in T_i is x_i, without decreasing the probability of simultaneous agreement.

For $b \in B$, let f_i^b denote the function obtained from f_i by fixing $x_1^i = b$. Also, let D_i^b be the distribution on the set $X_i - x_i$ obtained from D_i by conditioning on $x_1^i = b$.

Now, for $\mathbf{b} = (b_1, b_2, \ldots, b_j) \in B^j$ let $A(\mathbf{b})$ denote the event that $(\tilde{\alpha}_1(x_1) = b_1) \wedge \ldots \wedge (\tilde{\alpha}_j(x_j) = b_j)$. We can write the probability that all of the T_i compute correctly by conditioning on $\mathbf{b}$ as follows:

$$\sum_{\mathbf{b}\in B^{[j]}} \mathbf{Prob}[A(\mathbf{b})]\mathbf{Prob}[C([k])|A(\mathbf{b})] \tag{1}$$

We seek to upper bound this expression by:

$$\prod_{i=1}^{k} q_{D_i}(f_i, \mathcal{T}_{d_i}). \tag{2}$$

To do this we show:

Claim. For each $\mathbf{b} \in B^{[j]}$, the conditional probability of $C([k])$ given $A(\mathbf{b})$ is at most:

$$(\prod_{i=1}^{j} q_{D_i^{b_i}}(f_i^{b_i}, \mathcal{T}_{d_i}))(\prod_{i=j+1}^{k} q_{D_i}(f_i, \mathcal{T}_{d_i}))$$

Assuming the claim for the moment, we can then substitute into the expression (1) to obtain the following bound on the probability that all of the trees are correct:

$$(\prod_{i=j+1}^{k} q_{D_i}(f_i, \mathcal{T}_{d_i})) \sum_{\mathbf{b}\in B^j} \mathbf{Prob}[A(\mathbf{b})](\prod_{i=1}^{j} q_{D_i^{b_i}}(f_i^{b_i}, \mathcal{T}_{d_i})) \tag{3}$$

The sum can be rewritten as:

$$\sum_{\mathbf{b}\in B^j} \prod_{i=1}^{j} \mathbf{Prob}_{D_i}[x_i = b_i] q_{D_i^{b_i}}(f_i^{b_i}, \mathcal{T}_{d_i})),$$

which is equal to:

$$\prod_{i=1}^{j} \left(\mathbf{Prob} D_i[\tilde{\alpha}(x_i) = -1] q_{D_i^{-1}}(f_i^{-1}, \mathcal{T}_{d_i}) + \mathbf{Prob} D_i[\tilde{\alpha}(x_i) = 1] q_{D_i^{1}}(f_i^{1}, \mathcal{T}_{d_i})\right)$$

Now, the i^{th} term in this product corresponds to the probability of correctly computing f_i if we first probe x_i and then, depending on the outcome, use the optimal depth $d_i - 1$ tree to evaluate the residual function. Thus, we can upper bound this term by $p(f_i, D_i, d_i)$. But then, the expression (3) is upper bounded by the expression (2) as required.

So it suffices to prove the claim. Define $f_i^{A(\mathbf{b})}$ to be the function $f_i^{b_i}$ for $i \leq j$ and to be f_i otherwise. Similarly, the distribution $D_i^{A(\mathbf{b})}$ is equal to $D_i^{b_i}$ for $i \leq j$ and to

D_i otherwise. Observe that by the mutual independence of $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_k$, their joint distribution given $A(\mathbf{b})$ is the product distribution of $D_i^{A(\mathbf{b})}$ for i between 1 and k.

Let $T_i^{A(\mathbf{b})}$ be the tree obtained from contracting T_i under the assumption that $A(\mathbf{b})$ holds. Then the conditional probability that $T_i = f_i(x^i)$ for all i, given $A(\mathbf{b})$ is equal to the probability (with respect to the product distribution on $D_i^{A(\mathbf{b})}$) that for all i, $T_i^{A(\mathbf{b})} = f_i^{A(\mathbf{b})}$. Now for each i the depth of $T_i^{A(\mathbf{b})}$ has at most $d_i - 1$ if $i < j$, and is at most d_i for $i > j$, so we may apply induction to say that the probability with respect to the product distribution on $D_i^{A(\mathbf{b})}$ that for all i, $T_i^{A(\mathbf{b})} = f_i^{A(\mathbf{b})}$ is at most:

$$(\prod_{i=1}^{j} q_{D_i^{A(\mathbf{b})}}(f_i^{A(\mathbf{b})}, \mathcal{T}_{d_i}))(\prod_{i=j+1}^{k} q_{D_i^{A(\mathbf{b})}}(f_i^{A(\mathbf{b})}, \mathcal{T}_{d_i}))$$

which is equal to the expression in the claim. This proves the claim and the Theorem.

Remark. The proof of the Theorem can be extended to a more general model of decision tree computation. For this model, in the case of a single function we are given a function f from an arbitary domain S to R, and want to compute $f(s)$ for an unknown input $s \in S$. We are further given a set Q of admissible queries, where each query $q \in Q$ is a partition of S into sets $(S_1^q, S_2^q, \ldots S_r^q)$. The response to query q is the index i such that $s \in S_i^q$. The nodes of a decision tree are labeled by queries, and the branches out of the node correspond to the answers to the query. For a collection of functions f_i on disjoint domains S_i, the formulation of the product problem generalizes to this model. The statement and proof of the Theorem now go through assuming: (1) That the any allowed query depends only on variables from one function and (2) The distributions D_i are independent.

4 Help Bits

In the help bits problem, We have k boolean functions $f_1, f_2, \ldots, f_k$ over disjoint variable sets $X_1, X_2, \ldots, X_k$, Given an unknown assignment α to the variables of the set $X = X_1 \cup \ldots \cup X_k$, we want to evaluate $f_i(\alpha_i)$ for all i, by a decision forest. We are allowed to ask, "for free", an *arbitrary* set of l binary questions about the assignment α. The answer to these l questions is a vector $\mathbf{a} \in B^l$. For each such $\mathbf{a}$ we will have a decision forest $F^{\mathbf{a}} = (T_1^{\mathbf{a}}, T_2^{\mathbf{a}}, \ldots, T_k^{\mathbf{a}})$, where we require that $F^{\mathbf{a}}(\alpha)$ agrees with $(f_1(\alpha_1), \ldots, f_k(\alpha_k))$ for every assignment α that is consistent with $\mathbf{a}$.

Thus, such an algorithm is specified by l arbitrary boolean functions $h_1, h_2, \ldots, h_l$ (the help bits) on variable set X, together with 2^l decision forests. The complexity of the algorithm is the maximum depth of any of the $2^l k$ decision trees in these forests. In general, the decision tree $T_i^{\mathbf{a}}$ that computes $f_i(\alpha_i)$ for α consistent with $\mathbf{a}$ is allowed to probe variables outside of X_i. This is conceivably useful, because together with the help bits, such probes could imply information about the variables in X_i. For instance if one of the help bit functions is $(f_i(\alpha_i) \times \alpha_j(x))$ where x is a variable in X_j, then by probing the variable x, we can deduce $f_i(\alpha_i)$. If $T_i^{\mathbf{a}}$ only probes variables in X_i we say that it is *pure*. If each of the $2^l k$ decision trees are pure, the algorithm is pure.

In this paper, we will restrict attention to the case that, for some variable set X and boolean function f over X, each of the X_i are copies of X and the functions f_i are copies of f. The help bits problem $H^{k,l}(f)$ is to evaluate k copies of f given l help bits. Define $C^{k,l}(f)$ to be the complexity of the optimal algorithm that solves it. We also define the problem $H^{k,l}_{pure}(f)$ to be the same as $H^{k,l}(f)$ except that we require that the algorithm be pure. Define $C^{k,l}_{pure}(f)$ to be the complexity of the optimal pure algorithm. Our goal is to obtain bounds on $C^{k,l}(f)$ and $C^{k,l}_{pure}(f)$. The main result of this section (which is a slight refinement of Theorem 2), is:

Theorem 4.1 *For any boolean function f on n variables and any positive integer k,*

$$C^{k,k-1}_{pure}(f) \geq C^{k,k-1}(f) \geq deg^s(f).$$

If k is sufficiently large, then

$$C^{k,k-1}(f) = C^{k,k-1}_{pure}(f) = deg^s(f).$$

We first reformulate the problems $H^{k,l}(f)$ and $H^{k,l}_{pure}(f)$. Given functions $f_1, f_2, \ldots, f_k$ as above, and a decision forest F, we say that F *covers* the assignment α of X, with respect to $f_1, f_2, \ldots, f_k$, if $F(\alpha) = (f_1(\alpha), f_2(\alpha), \ldots, f_k(\alpha))$. Let $\tau^k(f, d)$ be the minimum number of forests, each consisting of trees of depth at most d, needed to cover all inputs with respect to f. Let $\tau^k_{pure}(f, d)$ be the corresponding minimum when we restrict to forests that are pure.

Proposition 4.1 *Let f be a boolean function and k, l, d be nonnegative integers. Then:*

1. *$C^{k,l}(f) \leq d$ if and only if $\tau^k(f, d) \leq 2^l$,*
2. *$C^{k,l}_{pure}(f) \leq d$ if and only if $\tau^k_{pure}(f, d) \leq 2^l$.*

In other words, $\lceil \log_2 \tau^k(f,d) \rceil$ is the minimum l such that $H^{k,l}$ can be solved with trees of depth d, and $\lceil \log_2 \tau^k_{pure}(f,d) \rceil$ is the minimum l such that $H^{k,l}_{pure}$ can be solved with trees of depth d.

Proof. We prove the first assertion; the proof of the second is completely analogous. If $C^{k,l}(f) \leq d$, then the 2^l forests given by the algorithm are also a cover and $\tau^k(f,d) \leq 2^l$. Now suppose $\tau^k(f,d) \leq 2^l$. Then there is a collection of 2^l forests that cover all assignments of X. Index these forest as $F^{\mathbf{z}}$ where $\mathbf{z}$ ranges over B^l. Order the forests lexicographically, and define $A(\mathbf{z})$ to be the set of assignments that are covered by $F^{\mathbf{z}}$ but not covered by $F^{\mathbf{y}}$ for any $\mathbf{y} \leq \mathbf{z}$. Then the sets $\{A(\mathbf{z}) : \mathbf{z} \in B^k\}$ partition the set of all assignments of X. Now define the help bit functions $h_1, h_2, \ldots, h_l$ so that for each α, $(h_1(\alpha), h_2(\alpha), \ldots, h_k(\alpha))$ is the unique index $\mathbf{z}$ such that $\alpha \in A(\mathbf{z})$. Then these functions together with $\{F^{\mathbf{z}} : \mathbf{z} \in B^l\}$ solves $H^{k,l}$. ∎

So we now concentrate on obtaining bounds on $\tau^k(f,d)$ and $\tau^k_{pure}(f,d)$. For this we need yet another definition. A *randomized (X,R)-decision tree algorithm* is a probability distribution Q over (X,R)-decision trees. Such an algorithm is said to *approximate f with probability p* if for each assignment α, if $\tilde{T}$ is a random decision tree chosen according to Q, then the probability that $\tilde{T}(\alpha) = f(\alpha)$ is at least p. We define $p(f,d)$ to be the maximum p such that there is a distribution Q over the set of decision trees of depth at most d that approximates f with probability p. It is easy to see that $p(f,d) \geq 1/2$. and that if $d = C(f)$, the ordinary decision tree complexity of f, then $p(f,d) = 1$. The following result relates $\tau(f,d)$ to $p(f,d)$.

Lemma 4.1 *For any boolean function f on n variables and $k, d \geq 0$, we have:*

$$\frac{1}{p(f,d)^k} \leq \tau(f,d) \leq \tau_{pure}(f,d) \leq \lceil \frac{nk \ln 2}{p(f,d)^k} \rceil$$

Proof. The middle inequality is trivial. For the last inequality, we use a standard probabilistic argument to show that there is family of at most $\lceil \frac{k \ln 2}{p(f,d)} \rceil$ pure forests of depth at most d that cover all of the assignments. Let Q be the distribution over (Y,R)-decision trees of depth at most d that approximates f with probability $p(f,d)$. For $i \leq k$, let Q_i be the corresponding distribution over the set of (X_i, R) decision trees; Q_i approximates f_i with probability $p(f,d)$. Consider the distribution $P = Q_1 \times \ldots \times Q_k$ over forests. Suppose we select t forests $\tilde{F}_1, \tilde{F}_2, \ldots, \tilde{F}_t$ according to P. For a given assignment α and $j \leq t$, the probability that $\tilde{F}_j$ covers α is at least $p(f,d)^k$. Thus the probability that none of the forests cover α is at most $(1-p(f,d))^t$, and the probability that there exists an assignment α that is covered by none of the forests is at most $2^{nk}(1-p(f,d)^k)^t < 2^{nk}e^{-p(f,d)^k t}$. If $t = \lceil (nk \ln 2)/p(f,d)^k \rceil$ then this expression is at most 1, so there is a positive probability that the forest covers all assignments, and so there must be a collection of t forests of depth d that cover all assignments.

Now we turn to the lower bound on $\tau(f,d)$. For this, we need the following relationship between $p(f,d)$ and the agreement probability $q_{\hat{D}}(f,d)$ with respect to a particular distribution $\hat{D}$ on assignments.

Lemma 4.2 *For any (Y,R)-boolean function f and integer $d \geq 0$, there exists a distribution $\hat{D}$ on assignments to Y such that $q_{\hat{D}}(f,d) = p(f,d)$.*

This is a variant of a fundamental observation of Yao [Y1], and follows from the min-max theorem for two person zero sum games.

Let $\hat{D}$ be the distribution of the lemma. Suppose that $F_1, F_2, \ldots, F_t$ is a family of forests that cover all assignments α to X. Consider the distribution P over all assignments α which is the product $\hat{D}_1 \times \hat{D}_2 \times \ldots \times \hat{D}_k$, where $\hat{D}_i$ is the copy of $\hat{D}$ on X_i. Then, by Theorem 3.1, for any forest F_i, the probability that it covers $\tilde{\alpha}$ is at most $p(f,d)^k$. Then the expected number of assignments covered by $F_1, F_2, \ldots, F_t$ is at most $tp(f,d)^k$. Since $F_1, F_2, \ldots, F_k$ covers all assignments, this expectation must be at least 1, so $t \geq 1/p(f,d)^k$. ∎

As an immediate corollary of the above lemma and proposition 4.1 we get the following bounds on the complexity of the help bits problem:

Corollary 4.1 *For any boolean function f on n variables and integers $k, l, d \geq 0$:*

1. *If $2^l \leq 1/p(f,d)^k$ then $C^{k,l}(f) > d$.*
2. *If $2^l \geq nk/p(f,d)^k$ then $C^{k,l}_{pure}(f) \leq d$.*

Next we need to connect the quantity $p(f,d)$ to the sign-degree $deg^s(f)$.

Proposition 4.2 *For any boolean function f, $p(f,d) > 1/2$ if and only if $d > deg^s(f)$.*

Proof. Let $d > deg^s(f)$. Then there is an n-variate polynomial $p(x_1, x_2, \ldots, x_n)$ of degree at most d such that $g(\alpha) > 0$ if and only if $f(\alpha) = 1$. By shifting the polynomial by a small constant we may assume that $g(\alpha)$ is never 0. We may assume without loss of generality that the sum of the absolute values of the coefficients of g is 1. Consider the following randomized decision tree algorithm: choose a monomial of g at random, where the probability a given monomial is chosen is the absolute value of its coefficient. Probe the variables of the monomial and output the product of the values. It is easily seen that for any assignment α, the probability of correctly evaluating $f(\alpha)$ minus the probability of incorrectly evaluating $f(\alpha)$ is equal to $|g(\alpha)| > 0$ (here we use that our domain is $\{-1, 1\}$). Thus for any α this algorithm correctly evaluates $f(\alpha)$ with probability exceeding $1/2$.

Now suppose $p(f, d) > 1/2$. There must exist a randomized decision tree algorithm Q on depth d trees that evaluates $f(\alpha)$ correctly with probability exceeding $1/2$. Now, it is well known, and easy to see (by induction on d, looking at the two subtrees of the root) that if T is a decision tree of depth d on variables $\{x_1, \ldots, x_n\}$ then there is a polynomial $g_T(x_1, \ldots, x_n)$ of degree d such that $g_T(\alpha) = T(\alpha)$ for all assignments α. Define the polynomial $g(x_1, \ldots, x_n)$ to be the sum of $Q(T)(g_T - 1/2)$ where the sum is over all trees of depth d and $Q(T)$ is the probability that T is selected under the distribution Q. Then $g(\alpha_1, \ldots, \alpha_n) = \mathbf{Prob}_Q[\tilde{T}(\alpha) = 1] - 1/2$. By the choice of Q, this latter term is positive if and only if $f(\alpha) = 1$. ∎

Theorem 4.1 now follows easily.

Proof of Theorem 4.1. By Corollary 4.1, $C^{k,k-1}(f) \leq deg^s(f)$ would follow from $2^{k-1} \geq nk/p(f, deg^s(f))^k$. This holds for all sufficiently large k since $p(f, deg^s(f)) > 1/2$, by Proposition 4.2.

Also, by Corollary 4.1, to show $C^{k,k-1}_{pure}(f) > deg^s(f) - 1$, it suffices to show $2^{k-1} < 1/p(f, deg^s(f) - 1)^k$ for all k, which follows immediately from the fact, by Proposition 4.2, that $p(f, deg^s(f) - 1) = 1/2$. ∎

Remark 1. It is interesting to note that, for k large enough, it is possible to construct to obtain an optimal algorithm in which all of the decision trees have a particularly simple form. The randomized algorithm in the proof of Proposition 4.2 uses only decision trees that correspond to computing monomials of g. Using this randomized algorithm in the proof of the upper bound of lemma 4.1 the decision trees used in the help-bits algorithm are all of the same form.

Remark 2. As noted in the introduction, if f is the majority function the $deg^s(f) = 1$

and so the decision trees used in the optimal algorithm for $H^{k,k-1}$ for large k all have depth 1. In the case that f is the majority function on three variables, Manuel Blum gave the following constructive protocol to solve $H^{k,k-1}$. Enumerate the subsets of $[k]$ having size at least $2k/3$. The number of these sets is 2^{ck} for some $c < 1$. Fix an encoding of these sets by ck bits. Now given k separate inputs to the majority-of-3 function, and imagine the inputs arranged in a $k \times 3$ array. In each row, at least two of the three entries agree with the majority value, so there is a column in which at least $2k/3$ of the entries agree with the function value on that column. For the help bits, we ask for the lowest index of such a column (requiring 2 bits) and then the set S of rows for which this column gives the function value (requiring ck bits.) Armed with this information, the value of the function on row r is equal to the entry in that row and the designated column if $r \in S$ and is the negative of the entry otherwise.

Remark 3. In the proof of the lower bound in Lemma 4.1, we used Theorem 3.1 in order to deduce that for any forest F of depth at most d, the probability with respect to a particular distribution P on assignments F is correct for all k functions is at most $p(f,d)^k$. In the special case $d = deg^s(f)-1$, which is the relevant case for proving that $C^{k,k-1} > deg^s(f) - 1$ in Theorem 4.1, there is an alternative argument. We sketch this argument, which has the benefit that it extends to other models besides decision trees, as will be seen in the next section. As noted above, for $d = deg^s(f) - 1$, we have $p(f,d) = 1/2$, and thus for $\tilde{\alpha}$ selected from $\hat{D}$ (the distribution of Lemma 4.2) any decision tree of depth d agrees with f with probability exactly $1/2$. In particular, this can be shown to imply that if we fix the values of any d variables then either that partial assignment occurs with probability 0 under $\hat{D}$, or that the value of f conditioned on this assignment is unbiased.

Now, define the random variable c_i to be 0 if $T_i(\tilde{\alpha}) = f_i(\tilde{\alpha})$ and 1 otherwise. We want to show that the probability that $c_i = 0$ for all i is at most $1/2^k$. In fact, the distribution on $(c_1, c_2, \ldots, c_k)$ is uniform on $\{0,1\}^k$. By the XOR lemma of [Vaz] (see also [CGHFRS]) a distribution over $\{0,1\}^k$ is uniform if for any subset J of $[k]$, the random variable c_J defined to be the XOR of the c_i for $i \in J$ is unbiased. Let s_J be the probability that $c_J = 0$. The event $c_J = 0$ is the same as the event that $T_J(\tilde{\alpha})(= \prod_{i\in J} T_i(\tilde{\alpha}))$ is equal to $f_J(\tilde{\alpha})(= \prod_{i \in J} f_i(\tilde{\alpha}))$. Now by combining the decision trees $\{T_i | i \in J\}$ we can get a single decision tree of depth at most $|J|d$ that computes T_J. We claim that such a decision tree must agree with f_J with probability exactly $1/2$, which is enough to finish the argument. We prove the claim by showing that for each leaf of the tree T_J that is reached with nonzero probability, $f_J(\tilde{\alpha})$ conditioned on $\tilde{\alpha}$ reaching the leaf is unbiased. For each such leaf of the tree, there is an $i \in J$

such that at most d variables of X_i appear on the path. Recall that the value of f_i is unbiased when conditioned on the values of these d variables. If we further condition the value of f_J by the values of all variables not in X_i, then f_i is still unbiased and therefore so is f_J.

Remark 4. One implication of Theorem 4.1 is that for large enough k, the best algorithm for $H^{k,k-1}(f)$ uses pure trees. It is reasonable to speculate that this is the case for $H^{k,l}(f)$ for all k and l, and this is open. For the case $k = 2$, it is interesting to note that for the case $k = 2$ and $l = 1$, it is not hard to show that pure tree algorithm can not do better than $C(f)$, the ordinary decision tree complexity of f. To see this, note that the help bit partitions the set of assignments of $X = X_1 \cup X_2$ into two groups A_1 and A_2. It is not hard to see that either the set of assignments on X_1 induced by A_1 is all of B^{X_1}, or the set of assignments on X_2 induced by A_2 must be all of B^{X_2}. In the first case, then given A_1, a pure tree computation for f on X_1 is as hard as the problem without the help bits, and in the second case, then given A_2, a pure tree computation for f on X_2 is as hard as the problem without the help bits.

5 Other Models

Some of the ideas used so far are also relevant to other models of computation. We can get results for these models that are similar to but neither as precise or as strong as what we obtain for descision trees. It is convenient to describe our results in the following very general framework. We fix some computational model for computing a function f on input $\alpha \in X$, and some class, $FEAS$, of "feasable" algorithms.

Our results will only hold for classes having certain closure properties. A class $FEAS$ is *closed under k-counting* if for any k algorithms in $FEAS$, any algorithm that runs all k of these algorithms on the input and accepts or rejects based on the number of computations out of k that accept, is also in $FEAS$. Examples of such classes are polynomial size circuits, which are closed under poly-counting, and polylog-bit communication complexity protocols which are closed under polylog-counting.

¿From such a class we define when a multi-input algorithm is feasable. An algorithm for computing a function f on a pair of inputs $\alpha_1, \alpha_2 \in D^2$ is said to be *rectangularly-feasable*, in $FEAS_*$, if for every fixed value of α_1 the induced algorithm for f is in $FEAS$, and for every fixed value of α_2 the induced algorithm for f is in $FEAS$. Notice that for the two examples mentioned above (and essentially any model one may think of), $FEAS \subset FEAS_*$. Thus, for example, for the case of poly-size circuits, the lower bounds given below for two-input algorithms apply to all poly-size circuits as well.

5.1 Products

A product theorem in such a setting may be proven using Yao's XOR-lemma [Y2], which we observe applies in this general setting. Let D_1, D_2 distributions, and denote $p_1 = q_{D_1}(f_1; FEAS)$, $p_2 = q_{D_2}(f_2; FEAS)$.

Lemma 5.1 *(Yao) Assume that $FEAS$ is closed under k-counting. Then*

$$q_{D_1 \times D_2}(f_1(\alpha_1) \oplus f_2(\alpha_2); FEAS_*) \leq p_1 p_2 + (1-p_1)(1-p_2) + 1/k^{\Omega(1)}$$

¿From this one can deduce an "approximate product theorem".

Theorem 5.1 *Assume that $FEAS$ is closed under k-counting. Then*

$$q_{D_1 \times D_2}(f_1, f_2; FEAS_*) \leq p_1 p_2 + 1/k^{\Omega(1)}.$$

Proof. Fix an algorithm A in $FEAS_*$, and denote by p_{YY} the probability that it is correct on both inputs, by p_{NN} the probability that it is incorrect on both, by p_{YN} the probability that it is correct only on the first input and by p_{NY} the probability that it is correct only on the second input. Since for every fixed value of α_1 the probability that A is correct on f_2 is at most p_2, then by averaging over all α_1, we have $p_{YY} + p_{NY} \leq p_2$. Similarly, $p_{YY} + p_{YN} \leq p_1$. Finally, Yao's xor-lemma implies $p_{YY} + p_{NN} \leq p_1 p_2 + (1-p_1)(1-p_2) + 1/k^{\Omega(1)}$. These inequalities, together with the fact that $p_{YY} + p_{YN} + p_{NY} + p_{NN} = 1$, directly imply $p_{YY} \leq p_1 p_2 + 1/k^{\Omega(1)}$, which proves the lemma. ∎

5.2 Help Bits

We can use the approximate product theorem to get help-bit results for randomized algorithms. Given a class of "feasable algorithms" $FEAS$, We say that a function is *randomly feasably computable*, in RFC, if there exists a probability distribution on algorithms in $FEAS$ such that for any input, an algorithm chosen from this distribution will be be correct on f with probability of at least $2/3$. The constant $2/3$ is not important as the usual "amplification" lemmas work in this general case.

Lemma 5.2 *If $FEAS$ is closed under k-counting then the constant $2/3$ can be replaced by $1/2 + 1/k$ (or by 2^{-k}) without changing the class RFC.*

For the case where $FEAS$ is the class of polynomial size circuits, it is known that randomization does not increase power, and thus RFC is exactly equal to the functions computable by deterministic poly-size circuits. For the case where $FEAS$ is polylog-bit communication protocols, RFC is the functions computable by randomized polylog-bit protocols with two-sided error.

Let us define what is feasable computation with a help-bit. Let $FEAS$ be a given class of algorithms. A 1-help-bit-feasable algorithm, in $FEAS^1$, is a set of two algorithms A_0, A_1 in $FEAS$, and a boolean function h, whose value on input α is the output of $A_{h(\alpha)}$. A function is in RFC^1 if there is a $FEAS^1_*$ algorithm for computing two copies of f, which on every pair of inputs is correct on both with probability of at least $2/3$. We then can prove a randomized help-bit theorem.

Theorem 5.2 *If $FEAS$ is closed under $O(1)$-counting then $RFC^1 = RFC$.*

Proof. Assume that $f \notin RFC$ then, amplifying and similarly to lemma 4.2, there exists a distribution D such that $q_D(f; FEAS) \leq 0.51$. Using the approximate product theorem, any $FEAS_*$ algorithm for two copies of f can be correct on at most $0.51^2 + o(1)$ fraction of inputs (under distribution $D \times D$). If follows that any $FEAS^1_*$ algorithm can be correct with probability at most twice that, a probability smaller than $2/3$ (again probability taken over a pair of inputs chosen from $D \times D$.) This in turn implies that $f \notin RFC^1$. ∎

For the case of boolean circuits, this was proven in [ABG].

5.3 The $\log k$ Barrier

The "approximate product" theorem and the "randomized help-bit" theorem can be naturally generalized to up to $\log k$ functions where the family $FEAS$ is closed under k-counting. After that, these techniques break down. It is unknown for example whether a polynomial size circuit using n help-bits can compute $n + 1$ copies of function which doesn't have polynomial size circuits. One can show that in a black box model, alternatively, relative to a particular oracle, that the generalizations are *false* using $\omega(\log k)$ functions.

Acknowledgement. The authors have had many conversations with several people regarding this research. We would expecially like to acknowledge the contributions of Richard Beigel, Nati Linial, and Avi Wigderson.

References

[ABG] A. Amir, R. Beigel, W. Gasarch, *Some connections between bounded query classes and nonuniform complexity*, Proceedings 5th Conference on Structure in Complexity Theory , 1990.

[Bsh] N.H. Bshouty, *On the extended direct sum conjecture*, Proc. 21st ACM Symp. on Theory of Computing, 1989, pp. 177-185.

[Cai] J. Cai, *Lower bounds for constant depth circuits in the presence of help bits*, Proc. 30th IEEE Symp. on Foundations of Computer Science, 1989, pp. 532–537, 1989.

[CGHFRS] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, R. Smolensky, *The bit extraction problem of t-resilient functions*, Proc. 26th IEEE Symp. on Foundations of Computer Science, 1985, 396-407.

[FKN] T. Feder, E. Kushilevitz, M. Naor, *Amortized Communication Complexity*, Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 239–248.

[HW] J. Hastad, A. Wigderson, *Composition of the Universal Relation*, in "Advances in Computational Complexity Theory", AMS-DIMACS book series, to appear.

[J] J. Ja'Ja', J. Takche *On the Validity of the Direct Sum Conjecture*, SIAM J. Comput.**15** (4), 1986, pp. 1004–1020.

[KKN] M. Karchmer, E. Kushilevitz, N. Nisan, *Fractional Covers and Communication Complexity*, in Proc. 7th Structures in Complexity Theory Conference, 1992, pp. 262–274.

[KRW] M. Karchmer, R. Raz, A. Wigderson, *On Proving Super-Logarithmic Depth Lower Bounds via the Direct Sum in Communication Complexity*, Proc. 6th Conference on Structures in Complexity Theory, 1991, pp. 299-304.

[IRW] R. Impagliazzo, R. Raz, A. Wigderson, *A Direct Product Theorem*, Proc. 9th IEEE Conference on Structure in Complexity Theory, 1994, to appear.

[Lin] Nathan Linial, personal communication.

[NW] N. Nisan, A. Wigderson, *Rounds in Communication Complexity Revisited*, SIAM Journal on Computing, **22**, (1) 1993.

[Vaz] U. Vazirani, *Randomness, adversaries and computation*, Ph.D. Thesis, UC Berkeley, 1986.

[Y1] A. Yao, *Theory and applications of trapdoor functions*, Proc. 23rd Annual IEEE Symp. on Foundations of Computer Science, 1982, 80-91.

[Y2] A. Yao, *Probabilistic computations: towards a unified measure of complexity*, Proc. 18th Annual IEEE Symp. on Foundations of Computer Science, 1977, 222-227.