Nearly Optimal Separations Between Communication (or Query) Complexity and Partitions

Andris Ambainis[1], Martins Kokainis[2], and Robin Kothari[3]

1 **Faculty of Computing, University of Latvia, Riga, Latvia**
`andris.ambainis@lu.lv`
2 **Faculty of Computing, University of Latvia, Riga, Latvia**
`martins.kokainis@lu.lv`
3 **Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, USA**
`rkothari@mit.edu`

Abstract

We show a nearly quadratic separation between deterministic communication complexity and the logarithm of the partition number, which is essentially optimal. This improves upon a recent power 1.5 separation of Göös, Pitassi, and Watson (FOCS 2015). In query complexity, we establish a nearly quadratic separation between deterministic (and even randomized) query complexity and subcube partition complexity, which is also essentially optimal. We also establish a nearly power 1.5 separation between quantum query complexity and subcube partition complexity, the first superlinear separation between the two measures. Lastly, we show a quadratic separation between quantum query complexity and one-sided subcube partition complexity.

Our query complexity separations use the recent cheat sheet framework of Aaronson, Ben-David, and Kothari. Our query functions are built up in stages by alternating function composition with the cheat sheet construction. The communication complexity separation follows from "lifting" the query separation to communication complexity.

1998 ACM Subject Classification F1.2 Modes of Computation

Keywords and phrases Query Complexity, Communication Complexity, Subcube Partition Complexity, Partition Bound

Digital Object Identifier 10.4230/LIPIcs.CCC.2016.4

1 Introduction

Deterministic communication complexity

In the standard model of communication complexity, we wish to compute a function $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, where the inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are given to two different players, while minimizing the communication between the players. We use $D^{\text{cc}}(F)$ to denote the deterministic communication complexity of F, the number of bits communicated in the worst case by the best deterministic protocol for the function F.

The partition number of F, denoted $\chi(F)$, is the least number of monochromatic rectangles in a partition or disjoint cover of $\mathcal{X} \times \mathcal{Y}$ (where a monochromatic rectangle is a set $A \times B$, with $A \subseteq \mathcal{X}$ and $B \subseteq \mathcal{Y}$, such that F takes the same value on all elements of $A \times B$). Yao [30] observed that any C-bit communication protocol for F partitions the set of all inputs $\mathcal{X} \times \mathcal{Y}$ into at most 2^C monochromatic rectangles, which gives us $\log \chi(F) \leq D^{\text{cc}}(F)$. This

31st Conference on Computational Complexity (CCC 2016).
Editor: Ran Raz; Article No. 4; pp. 4:1–4:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 Known separations between deterministic communication complexity, $D^{cc}(F)$, and partition number, $\chi(F)$.

Separation	Reference
$D^{cc}(F) \geq 2\log\chi(F)$	[18]
$D^{cc}(F) = \widetilde{\Omega}\left(\log^{1.5}\chi(F)\right)$	[10]
$D^{cc}(F) \geq \left(\log\chi(F)\right)^{2-o(1)}$	Theorem 1.1
$D^{cc}(F) = O\left(\log\chi(F)^2\right)$ for all $F : \mathcal{X}\times\mathcal{Y}\to\{0,1\}$	

turns out to be a powerful lower bound, and in fact almost all lower bound techniques for deterministic communication complexity, including the partition bound, discrepancy, fooling sets, (nonnegative) rank, and various norm-based methods [13, 14, 22], actually lower bound $\log\chi(F)$.

In addition to being a fruitful lower bound technique, $\log\chi(F)$ also yields an upper bound on $D^{cc}(F)$. Aho, Ullman, and Yannakakis [4] showed that for all $F : \mathcal{X}\times\mathcal{Y}\to\{0,1\}$, we have

$$D^{cc}(F) = O(\log^2\chi(F)). \tag{1}$$

It has been a long-standing open problem to determine whether this upper bound can be improved (see, e.g., [19, Open Problem 2.10]). We show that the upper bound in (1) is essentially optimal.

▶ **Theorem 1.1.** *There exists a function $F : \mathcal{X}\times\mathcal{Y}\to\{0,1\}$ with $D^{cc}(F) \geq \left(\log\chi(F)\right)^{2-o(1)}$.*

Until recently, the best known separation between the two measures was only by a factor of 2 [18]. Recently, Göös, Pitassi, and Watson [10] showed that there exists a function F with $D^{cc}(F) = \widetilde{\Omega}(\log^{1.5}\chi(F))$, where the notation $\widetilde{\Omega}(m)$ hides $\mathrm{poly}(\log m)$ factors. Table 1 summarizes known separations between D^{cc} and $\log\chi$.

Deterministic query complexity

In the model of query complexity, we wish to compute a function $f : \{0,1\}^n\to\{0,1\}$ on an input $x\in\{0,1\}^n$ given query access to the bits of the input, i.e., we can only access the input via a black box that accepts an index $i\in[n]$ (where $[n] := \{1,2,\ldots,n\}$) and responds with $x_i\in\{0,1\}$. The goal is to compute $f(x)$ while minimizing the number of queries made to the black box. Let $D(f)$ to denote the deterministic query complexity of f, the number of queries made by the best deterministic algorithm that computes f correctly on all inputs.

As in communication complexity, most lower bounds for deterministic query complexity are based on the simple observation that any d-query algorithm computing f partitions the domain $\{0,1\}^n$ into at most 2^d monochromatic subcubes where each subcube fixes at most d variables. A subcube is a restriction of the hypercube where some variables have been fixed, and it is monochromatic if f takes the same value on all inputs in the subcube. This motivates defining the subcube partition complexity of f as a smallest d such that the domain $\{0,1\}^n$ can be partitioned into at most 2^d monochromatic subcubes that each fix at most d variables. Subcube partition complexity can also be viewed as an unambiguous version of certificate complexity as explained in Section 3, and hence we denote this measure $\mathrm{UC}(f)$.

Table 2 Known separations between deterministic query complexity and subcube partition complexity.

Separation	Reference
$D(f) = \Omega(\mathrm{UC}(f)^{1.261})$	[27]
$D(f) = \widetilde{\Omega}(\mathrm{UC}(f)^{1.5})$	[10]
$D(f) \geq \mathrm{UC}(f)^{2-o(1)}$	Theorem 1.2
$D(f) = O(\mathrm{UC}(f)^2)$ for all $f : \{0,1\}^n \to \{0,1\}$	

Table 3 Known separations between randomized query complexity and subcube partition complexity.

Separation	Reference
$R(f) = \Omega(\mathrm{UC}(f)^{1.058})$	[17]
$R(f) = \widetilde{\Omega}(\mathrm{UC}(f)^{1.5})$	[9]
$R(f) \geq \mathrm{UC}(f)^{2-o(1)}$	Theorem 1.3
$R(f) = O(\mathrm{UC}(f)^2)$ for all $f : \{0,1\}^n \to \{0,1\}$	

Due to the observation above, we have $\mathrm{UC}(f) \leq D(f)$. It turns out that this lower bound is also relatively tight: for all $f : \{0,1\}^n \to \{0,1\}$ we have

$$D(f) = O(\mathrm{UC}(f)^2). \tag{2}$$

We show that this upper bound is essentially optimal.

▶ **Theorem 1.2.** *There exists a total function f with $D(f) \geq \mathrm{UC}(f)^{2-o(1)}$.*

The first separation between these two measures was a power 1.261 separation by Savický, which was recently improved by Göös, Pitassi, and Watson [10] to power 1.5. Table 2 summarizes known separations between these measures.

Randomized query complexity

We can extend the query model to allow randomized algorithms in the natural way. We define the bounded-error randomized query complexity of a function f, $R(f)$, to be the minimum number of queries needed in the worst case by a randomized algorithm that outputs $f(x)$ on input x with probability at least $2/3$.

As before, almost all lower bound techniques for randomized query complexity are upper bounded by $\mathrm{UC}(f)$, as shown in [17]. This includes the partition bounds [13, 14], approximate polynomial degree [25], approximate nonnegative junta degree (also known as nonnegative literal degree or conical junta degree) [16], block sensitivity [24], randomized certificate complexity or fractional block sensitivity [1, 7, 29], and the classical analogue of the quantum adversary bound [20, 28, 2].

Since we obviously have $R(f) \leq D(f)$, using (2) we know that $R(f) = O(\mathrm{UC}(f)^2)$. We show that this upper bound is also essentially optimal.

▶ **Theorem 1.3.** *There exists a total function f with $R(f) \geq \mathrm{UC}(f)^{2-o(1)}$.*

The first asymptotic separation between these measures was a power 1.058 separation by Racicot-Desloges, Santha, and Kothari [17], which was later improved by Göös, Jayram, Pitassi, and Watson [9] to a power 1.5 separation. Table 3 summarizes the known separations between these measures.

Quantum query complexity

The query model can also be naturally extended to quantum algorithms. We denote by $Q(f)$ the bounded-error quantum query complexity of f, the minimum number of queries made in the worst case by a quantum algorithm that outputs $f(x)$ on input x with probability at least $2/3$. (See [6] for a formal definition.)

As before, since $Q(f) \leq D(f)$, using (2) we know that $Q(f) = O(\mathrm{UC}(f)^2)$. However, prior to our work no function was known for which $Q(f) \gg \mathrm{UC}(f)$ was known. Furthermore, the functions previously used to show separations between $D(f)$ or $R(f)$ and $\mathrm{UC}(f)$ do not separate $Q(f)$ from $\mathrm{UC}(f)$. Indeed, even the functions constructed to prove Theorem 1.2 and Theorem 1.3 do not separate $Q(f)$ from $\mathrm{UC}(f)$. Despite this, we give the first superlinear separation between $Q(f)$ and $\mathrm{UC}(f)$.

▶ **Theorem 1.4.** *There exists a total function f with $Q(f) \geq \mathrm{UC}(f)^{1.5-o(1)}$.*

We are also able to show an improved separation between quantum query complexity and one-sided subcube partition complexity, denoted by $\mathrm{UC}_1(f)$, which is similar to subcube partition complexity except that we only need to partition the 1-inputs using monochromatic subcubes.

For this measure, the quadratic upper bound $D(f) = O(\mathrm{UC}_1(f)^2)$ still holds [8, Proposition 5], and hence $Q(f) = O(\mathrm{UC}_1(f)^2)$. We show this upper bound is optimal up to log factors, qualitatively improving upon [10] and [9] who proved the same result for deterministic and randomized query complexity respectively.

▶ **Theorem 1.5.** *There exists a total function f with $Q(f) = \widetilde{\Omega}(\mathrm{UC}_1(f)^2)$.*

2 High-level overview

We now provide a high-level overview of the separations shown.

Deterministic communication complexity

We prove Theorem 1.1 by showing the analogous separation in query complexity (Theorem 1.2) and "lifting" the result to communication complexity, which is also the strategy used in [10]. Essentially, the deterministic simulation theorem of [10] provides a black-box way of converting a query separation between $D(f)$ and $\mathrm{UC}(f)$ to a separation between $D^{\mathrm{cc}}(F)$ and $\log \chi(F)$. The theorem weakens the separation by log of the input size of f, but with a suitable choice of parameters this is negligible compared to the $o(1)$ term in the separation.

Deterministic query complexity

To prove Theorem 1.2, we use the recently introduced cheat sheet framework [3] and the commonly used technique of function composition. Before describing the construction, we need to define some notation. For any functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, we define the composed function $f \circ g$ to be the function on mn bits whose output on $y = (y_{11}, \ldots, y_{1m}, \ldots, y_{n1}, \ldots, y_{nm})$ is $f(g(y_{11}, \ldots, y_{1m}), \ldots, g(y_{n1}, \ldots, y_{nm}))$. Let $\textsc{And}_n$ and $\textsc{Or}_n$ denote the And and Or function on n bits respectively. For any function f, we use f_{CS} to denote the "cheat sheet version" of f, a new total Boolean function constructed from f. (We review the cheat sheet framework in Section 4.)

An interesting feature of the cheat sheet framework is that $\mathrm{UC}_1(f_{\mathrm{CS}})$ can be substantially smaller than $\mathrm{UC}_1(f)$ because one can construct a partition for inputs with $f_{\mathrm{CS}} = 1$ without using a partition for inputs with $f = 1$. This property is crucial for our construction but is not sufficient by itself because the complexity of the best partition for inputs with $f_{\mathrm{CS}} = 0$ is of a similar order as $\mathrm{UC}(f)$. To deal with this, we combine the cheat sheet construction with several other steps which rebalance the complexity of partitions for $f = 1$ and $f = 0$.

We construct our function in stages starting with the function $f_0 = \textsc{And}_n$ that achieves no separation between $D(f)$ and $\mathrm{UC}(f)$. We then compose the function with $\textsc{Or}_n$, construct

the cheat sheet version, and then compose with AND_n, to obtain the function $f_1 = \text{AND}_n \circ (\text{OR}_n \circ \text{AND}_n)_{\text{CS}}$, which achieves a power 3/2 separation between $D(f)$ and $\text{UC}(f)$. Repeating this construction once more yields $f_2 = \text{AND}_n \circ (\text{OR}_n \circ \text{AND}_n \circ (\text{OR}_n \circ \text{AND}_n)_{\text{CS}})_{\text{CS}}$, which achieves a power 5/3 separation, and so on. The function f_k achieves a $(2k+1)/(k+1)$ separation, which yields a $2 - o(1)$ separation if we choose k to be a slow growing function of n.

Randomized query complexity

The function constructed above also yields the separation in Theorem 1.3 with slightly worse parameters. The analysis of the constructed function is similar since deterministic and randomized query complexities behave similarly with respect to the cheat sheet technique and with respect to composition with the AND and OR functions.

Quantum query complexity

Lastly, we establish the quantum separations using two functions introduced by Aaronson, Ben-David and Kothari [3]: the BLOCK-k-SUM-OF-k-SUMS function, which we denote BKK, and the BLOCK-k-SUM function, which we denote BK-SUM. The function BKK_{CS} yields the separation in Theorem 1.5. The separation in Theorem 1.4 requires a function constructed in stages again. The first function is $f_1 = \text{AND} \circ \text{BKK}_{\text{CS}}$, which achieves a power 5/4 separation, the next is $f_2 = \text{AND}_n \circ (\text{BK-SUM}_n \circ f_1)$, which achieves a power 4/3 separation and so on. The function f_k achieves a power $(3k+2)/(2k+2)$ separation.

3 Preliminaries

Communication complexity

The only communication complexity measures we need are $D^{\text{cc}}(F)$ and $\chi(F)$, which were defined in Section 1. The interested reader is referred to [19, 15] for more formal definitions of these measures.

Query complexity

For more formal definitions of measures introduced in Section 1, the reader is referred to the survey by Buhrman and de Wolf [6]. The only measure not covered in the survey is subcube partition complexity, which is explained in detail in [17].

Subcube partition complexity can also be viewed as unambiguous certificate complexity and we use this perspective in this paper. To explain this, let us begin with certificate complexity.

A certificate for an input $x \in \{0,1\}^n$ is a subset $S \subseteq [n]$ of indices and claimed values for these bits, such that x is consistent with the certificate and any input y consistent with the certificate satisfies $f(x) = f(y)$. In other words, a certificate for x is a partial assignment of bits consistent with x such that any other string consistent with this partial assignment has the same function value as x. For $b \in \{0,1\}$, the b-certificate complexity of f, denoted $C_b(f)$, is the size of the smallest certificate for x maximized over all inputs with $f(x) = b$. The certificate complexity of f, $C(f)$, is defined as $C(f) := \max\{C_0(f), C_1(f)\}$. Alternately, $C_1(f)$ is the smallest w such that f can be written as a width-w DNF, i.e., a DNF in which each term contains at most w variables. Similarly, $C_0(f)$ corresponds to CNF width.

Unambiguous certificate complexity is defined similarly, except we require the set of certificates to be unambiguous, i.e., at most one certificate from the set of all certificates should work for a given input. In other words, the unambiguous 1-certificate complexity of f is the minimum w such that f can be written as a width-w DNF in which at most one term evaluates to 1 on any input. Similar to certificate complexity, we denote unambiguous b-certificate complexity by $\mathrm{UC}_b(f)$ and define $\mathrm{UC}(f) := \max\{\mathrm{UC}_0(f), \mathrm{UC}_1(f)\}$. Clearly, since unambiguous certificates are more restricted than certificates, we have for $b \in \{0,1\}$, $C_b(f) \leq \mathrm{UC}_b(f)$ and $C(f) \leq \mathrm{UC}(f)$.

For example, consider the OR_n function on n bits defined as $\bigvee_{i\in[n]} x_i$. Clearly $C_0(\mathrm{OR}_n) = n$ since we must examine all n bits to be sure that all $x_i = 0$. On the other hand, $C_1(\mathrm{OR}_n) = 1$ since the location of any 1 in the input is a certificate. Obviously $\mathrm{UC}_0(\mathrm{OR}_n)$ remains n. However, a single 1 in the input is not an unambiguous 1-certificate since inputs with multiple 1s would have multiple valid certificates. In other words, although $\bigvee_{i\in[n]} x_i$ is a valid DNF representation of OR_n, it is not unambiguous since several terms can simultaneously be 1. So consider the following DNF:

$$\mathrm{OR}_n(x) = x_1 \vee \overline{x_1}x_2 \vee \overline{x_1x_2}x_3 \vee \cdots \vee \overline{x_1x_2\cdots x_{n-1}}x_n\,. \tag{3}$$

This DNF is unambiguous since any term evaluating to 1 prevents other terms from evaluating to 1. Thus we have $\mathrm{UC}_1(\mathrm{OR}_n) \leq n$. Although this result in trivial because $\mathrm{UC}_1(f) \leq n$ for any n-bit function f, this DNF representation of OR_n will be useful to us later later because it has the property that every unambiguous certificate has only one unnegated index x_i.

Composition theorems

Composition theorems relate the complexity of composed functions with the complexities of the individual functions. For example, for all Boolean functions f and g, $D(f\circ g) = D(f)D(g)$ [29, 23]. In our construction we will repeatedly compose functions with AND_n and OR_n, and hence we need to understand the complexities of the resulting functions.

▶ **Lemma 3.1** (AND/OR composition). *For any total Boolean function f, the following bounds hold:*

- $D(\mathrm{AND}_n \circ f) = nD(f)$
- $R(\mathrm{AND}_n \circ f) = \Omega(nR(f))$
- $Q(\mathrm{AND}_n \circ f) = \Omega(\sqrt{n}Q(f))$
- $C_0(\mathrm{AND}_n \circ f) \leq C_0(f)$
- $C_1(\mathrm{AND}_n \circ f) \leq nC_1(f)$
- $\mathrm{UC}_0(\mathrm{AND}_n \circ f) \leq \mathrm{UC}_0(f) + (n-1)\mathrm{UC}_1(f)$
- $\mathrm{UC}_1(\mathrm{AND}_n \circ f) \leq n\mathrm{UC}_1(f)$
- $D(\mathrm{OR}_n \circ f) = nD(f)$
- $R(\mathrm{OR}_n \circ f) = \Omega(nR(f))$
- $Q(\mathrm{OR}_n \circ f) = \Omega(\sqrt{n}Q(f))$
- $C_0(\mathrm{OR}_n \circ f) \leq nC_0(f)$
- $C_1(\mathrm{OR}_n \circ f) \leq C_1(f)$
- $\mathrm{UC}_0(\mathrm{OR}_n \circ f) \leq n\mathrm{UC}_0(f)$
- $\mathrm{UC}_1(\mathrm{OR}_n \circ f) \leq (n-1)\mathrm{UC}_0(f) + \mathrm{UC}_1(f)$

Proof. We prove the claims for the function $\mathrm{AND}_n \circ f$. Similar reasoning proves the analogous claims for the function $\mathrm{OR}_n \circ f$.

The first property follows from the fact that $D(f\circ g) = D(f)D(g)$ for any Boolean functions f and g [29, 23]. $R(\mathrm{AND}_n \circ f) = \Omega(nR(f))$ was recently proved by [9]. $Q(\mathrm{AND}_n \circ f) = \Omega(\sqrt{n}Q(f))$ because $Q(f \circ g) = \Theta(Q(f)Q(g))$ for any Boolean functions f and g [12, 26, 21] and we know that $Q(\mathrm{AND}_n) = Q(\mathrm{OR}_n) = \Theta(\sqrt{n})$ [11, 5].

We have $C_0(\mathrm{AND}_n \circ f) \leq C_0(f)$ since a 0-certificate for AND_n is a 0-input to it, which corresponds to an instance of f that evaluates to 0. On the other hand, by certifying that all n instances of f evaluate to 1, we can certify $\mathrm{AND}_n \circ f$ evaluates to 1, and hence $C_1(\mathrm{AND}_n \circ f) \leq nC_1(f)$.

We can unambiguously certify that $\text{AND}_n \circ f$ evaluates to 0 by unambiguously certifying the value of the first (from the left) 1-input to the AND_n gate and unambiguously certifying that all previous inputs are 0. This is the same idea used to construct the unambiguous DNF for OR_n in (3). This construction gives $\text{UC}_0(\text{AND}_n \circ f) \leq \text{UC}_0(f) + (n-1)\text{UC}_1(f)$. We can unambiguously certify that $\text{AND}_n \circ f$ evaluates to 1 by providing unambiguous 1-certificates for all n instances of f. This gives $\text{UC}_1(\text{AND}_n \circ f) \leq n\text{UC}_1(f)$. ◀

4 Cheat sheet framework

We now overview the recently introduced cheat sheet framework [3]. The framework as presented in [3] is more general and can fulfill different objectives such as making partial functions total. We present a restricted version of the framework that only works for total functions. We use the framework because it makes 1-certificates unambiguous in a natural way.

▶ **Definition 4.1** (Cheat sheet version of a total function)**.** Let $f : \{0,1\}^N \to \{0,1\}$ be a function, $c = 10 \log N$ and $m = 10C(f) \log^2 N$. Then the *cheat sheet version of* f, denoted f_{CS}, is a total function

$$f_{\text{CS}} : (\{0,1\}^N)^c \times (\{0,1\}^m)^{2^c} \to \{0,1\}.$$

Let the input be written as $(x^1, x^2, \ldots, x^c, Y_1, Y_2, \ldots, Y_{2^c})$, where for all $i \in [N]$, $x^i \in \{0,1\}^N$ and for all $j \in [2^c]$, $Y_j \in \{0,1\}^m$. Let $\ell_i = f(x^i)$ and $\ell \in [2^c]$ be the positive integer corresponding to the binary string $\ell_1 \ell_2 \ldots \ell_c$. Then we define the value of $f_{\text{CS}}(x^1, x^2, \ldots, x^c, Y_1, Y_2, \ldots, Y_{2^c})$ to be 1 if and only if Y_ℓ contains certificates for $f(x^i) = \ell_i$ for all $i \in [c]$.

Informally, the cheat sheet construction takes any total function f and converts it into a new total function f_{CS} in the following way. An input to the new function f_{CS} first contains $c = 10 \log N$ inputs to f and then a vast array of size 2^c of cells of size m bits. The outputs of these c inputs to f is a bit string $\ell_1 \ell_2 \ldots \ell_c$ of length c that represents an integer $\ell \in [2^c]$ in the natural way. We treat this integer ℓ as an address into this array of size 2^c and say that these c inputs to f point to the ℓ^{th} cell of the array. At the ℓ^{th} cell of the array we require certificates certifying that this was indeed the cell pointed to by the c inputs to f. In other words, we require certificates certifying that $f(x^i)$, the output of f acting on the i^{th} input, is indeed equal to ℓ_i for all $i \in [c]$. Since a certificate for a single f consists of $C(f)$ pointers to the input, a certificate is of size $C(f) \log N$ bits, and hence c certificates are of size $m = C(f)c \log N = 10C(f) \log^2 N$. The function f_{CS} is defined to be 1 if and only if the input satisfies this property, i.e., if the cell pointed to by the c instances does indeed contain certificates certifying it is the correct cell.

This construction preserves the complexity of f with respect to some measures. For example, $D(f_{\text{CS}})$ equals $D(f)$ up to log factors. The upper bound uses the natural algorithm for f_{CS}: the deterministic algorithm first computes the c copies of f on inputs x^1 to x^c and finds the cell pointed to by these c inputs. Then it checks if the certificates in this cell certify that this is the right cell. This requires $cD(f)$ queries to compute the c copies, m queries to read the contents of the cell and $cC(f)$ queries to check if the certificates are all correct. Overall this uses $O(cD(f))$ queries. We also have $D(f_{\text{CS}}) = \Omega(D(f))$, because intuitively if an algorithm cannot compute f it has no hope of finding the cheat sheet since that would require solving c copies of f or searching in an array of size n^{10}. Similarly, many measures behave as expected under cheat sheets, and we show this below.

▶ **Lemma 4.2** (Complexity of cheat sheet functions). *For any total function $f : \{0,1\}^N \to \{0,1\}$, if $f_{\mathrm{CS}} : \{0,1\}^{N'} \to \{0,1\}$ denotes the cheat sheet version of f as defined in Definition 4.1, then we have the following upper and lower bounds:*

- $D(f_{\mathrm{CS}}) = \Omega(D(f))$
- $R(f_{\mathrm{CS}}) = \Omega(R(f)/\log^2 N)$
- $Q(f_{\mathrm{CS}}) = \Omega(Q(f))$
- $C_0(f_{\mathrm{CS}}) = O(C(f)\log^2 N)$
- $C_1(f_{\mathrm{CS}}) = O(C(f)\log^2 N)$
- $\mathrm{UC}_0(f_{\mathrm{CS}}) = O(\mathrm{UC}(f)\log^2 N)$
- $\mathrm{UC}_1(f_{\mathrm{CS}}) = O(C(f)\log^2 N)$
- $N' = O(N^{12})$

Proof. We have $D(f_{\mathrm{CS}}) = \Omega(D(f))$ [3, Lemma 21], $R(f_{\mathrm{CS}}) = \Omega(R(f)/\log^2 N)$ [3, Lemma 6], and $Q(f_{\mathrm{CS}}) = \Omega(Q(f))$ [3, Lemma 12].

We have $C_0(f_{\mathrm{CS}}) = O(C(f)\log^2 N)$ because a valid 0-certificate for f_{CS} can first certify the c outputs to f, which requires $O(cC(f))$ queries. This points to a cell ℓ. The certificate can then contain the contents of cell ℓ of size $O(C(f)\log^2 N)$ and the locations pointed to (and the bits contained at these locations) by the certificates in cell ℓ. After querying this cell and all the locations pointed to by the certificates in this cell, it can be determined with no further queries if this cell is incorrectly filled. We have $C_1(f_{\mathrm{CS}}) = O(C(f)\log^2 N)$ since the location of the correct cell and the pointers within that cell along with the bits they point to forms a 1-certificate.

We have $\mathrm{UC}_0(f_{\mathrm{CS}}) = O(\mathrm{UC}(f)\log^2 N)$ using the same argument as for certificate complexity. We first certify the c outputs to f unambiguously using unambiguous certificates of size $\mathrm{UC}(f)$. This points to a cell ℓ. The certificate also contains the contents of cell ℓ and the locations pointed to (and the bits at these locations) by the certificates in cell ℓ. This certificate is unambiguous because this certificate evaluating to true prevents any other certificate from evaluating to true. To see this, note that if another certificate tries to certify a different value of ℓ then this will be an invalid certificate. If the certificate claims the same value of ℓ, then it must use the same certificates for the c instances of f because we used unambiguous certificates and hence there is only one valid certificate for each $f(x^i) = \ell_i$. Now if the other certificate has the same value of ℓ but different claimed values for the contents of the ℓ^{th} cell or the locations pointed to by the cell, this will be inconsistent with the actual input since our original certificate was consistent with the input.

We have $\mathrm{UC}_1(f_{\mathrm{CS}}) = O(C(f)\log^2 N)$. For this case an unambiguous certificate will contain only the contents of cell ℓ and the locations pointed to by the certificates in cell ℓ along with the bits contained at these locations. This is identical to the 1-certificate we constructed above. Since this is clearly a valid certificate, we only need to show it is unambiguous, i.e., that if this certificate evaluates to true, all other certificates must fail. If another certificate has a different value of ℓ, then its contents will not be able to certify that the output of the c functions equals ℓ and the certificate will be rejected. On the other hand, if the other certificate has the same value of ℓ but different claimed values for the contents of the cell or the locations pointed to by the cell, this will be inconsistent with the input since our original certificate was consistent with the input.

Lastly, we need to upper bound the input size of f_{CS}. From Definition 4.1 we know the input size is $cN + m2^c = 10N\log N + 10N^{10}C(f)\log^2 N = O(10N^{11}\log^2 N) = O(N^{12})$. ◀

5 Randomized query complexity vs. subcube partitions and deterministic communication vs. partition number

Randomized query complexity vs. subcube partitions

We now establish the following theorem which implies Theorem 1.3, which in turn implies Theorem 1.2.

▶ **Theorem 5.1.** *For every $k \geq 0$, there exists a total Boolean function $f_k : \{0,1\}^{N_k} \to \{0,1\}$, such that $R(f_k) = \widetilde{\Omega}(n^{2k+1})$ and $\mathrm{UC}(f_k) = \widetilde{O}(n^{k+1})$. Hence there is a function f with $R(f) \geq \mathrm{UC}(f)^{2-o(1)}$.*

Proof. Let $f_0 = \textsc{And}_n$ and f_k be defined inductively as $f_k := \textsc{And}_n \circ (\textsc{Or}_n \circ f_{k-1})_{\mathrm{CS}}$, i.e., f_k is the function obtain by composing $\textsc{And}_n$ with the cheat sheet version of $\textsc{Or}_n$ composed with f_{k-1}.

We prove the claim by induction on k. The induction hypothesis and the base case, $f_0 = \textsc{And}_n$, are presented below, where N_k is the input size of the function f_k.

Induction hypothesis (f_k)
- $N_k = O(n^{25^k})$
- $D(f_k) = \widetilde{\Omega}(n^{2k+1})$
- $R(f_k) = \widetilde{\Omega}(n^{2k+1})$
- $C_0(f_k) = \widetilde{O}(n^k)$
- $C_1(f_k) = \widetilde{O}(n^{k+1})$
- $\mathrm{UC}_0(f_k) = \widetilde{O}(n^{k+1})$
- $\mathrm{UC}_1(f_k) = \widetilde{O}(n^{k+1})$

Base case ($f_0 = \textsc{And}_n$)
- $N_0 = n$
- $D(f_0) = n$
- $R(f_0) = \Omega(n)$
- $C_0(f_0) \leq 1$
- $C_1(f_0) \leq n$
- $\mathrm{UC}_0(f_0) \leq n$
- $\mathrm{UC}_1(f_0) \leq n$

The complexities of $f_0 = \textsc{And}_n$ are straightforward to show and also follow from the general composition lemma (Lemma 3.1) by letting f be the one-bit identity function. Clearly the base case is consistent with the induction hypothesis.

We now show that the induction hypothesis for f_k implies the same for f_{k+1}. First we upper bound the input size of $f_{k+1} = \textsc{And}_n \circ (\textsc{Or}_n \circ f_k)_{\mathrm{CS}}$. Since the input size of f_k is $O(n^{25^k})$, the input size of $\textsc{Or}_n \circ f_k$ is $O(n^{25^k+1})$ and the input size of $(\textsc{Or}_n \circ f_k)_{\mathrm{CS}}$ is $O(n^{12(25^k+1)})$ (from Lemma 4.2). Hence the input size of f_{k+1} is $O(n^{12(25^k+1)+1}) = O(n^{12(25^k)+13}) = O(n^{25^{k+1}})$.

The deterministic query complexity of f_{k+1} can be lower bounded as follows:

$$D(f_{k+1}) = D(\textsc{And}_n \circ (\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) = nD((\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) = \Omega(nD(\textsc{Or}_n \circ f_k)) = \widetilde{\Omega}(n^{2k+3}),$$

where we used Lemma 3.1 and Lemma 4.2 to compute the relevant measures. The same calculation also works for $R(f_{k+1})$ up to log factors since $R(f)$ and $D(f)$ behave similarly in the aforementioned lemmas up to log factors. Similarly using Lemma 3.1 and Lemma 4.2 we have

$$C_0(f_{k+1}) = C_0(\textsc{And}_n \circ (\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) \leq C_0((\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) = \widetilde{O}(C(\textsc{Or}_n \circ f_k)) = \widetilde{O}(n^{k+1})$$

and

$$C_1(f_{k+1}) = C_1(\textsc{And}_n \circ (\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) \leq nC_1((\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) = \widetilde{O}(nC(\textsc{Or}_n \circ f_k)) = \widetilde{O}(n^{k+2}).$$

In these bounds we do not differentiate between $\log N_k$ and $\log n$ because they are asymptotically equal, since $\log N_k = 25^k \log n = O(\log n)$. Finally, using Lemma 3.1 and

Lemma 4.2 again we have

$$\begin{aligned}
\mathrm{UC}_0(f_{k+1}) &= \mathrm{UC}_0(\textsc{And}_n \circ (\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) \\
&\leq \max\{\mathrm{UC}_0((\textsc{Or}_n \circ f_k)_{\mathrm{CS}}), n\mathrm{UC}_1((\textsc{Or}_n \circ f_k)_{\mathrm{CS}})\} \\
&= \widetilde{O}\left(\max\{\mathrm{UC}(\textsc{Or}_n \circ f_k), nC(\textsc{Or}_n \circ f_k)\}\right) = \widetilde{O}(n^{k+2}) \qquad \text{and} \\
\mathrm{UC}_1(f_{k+1}) &= \mathrm{UC}_1(\textsc{And}_n \circ (\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) \leq n\mathrm{UC}_1((\textsc{Or}_n \circ f_k)_{\mathrm{CS}}) \\
&= \widetilde{O}(nC(\textsc{Or}_n \circ f_k)) = \widetilde{O}(\max\{nC_0(f_k), C_1(f_k)\} = \widetilde{O}(n^{k+2}).
\end{aligned}$$

This completes the induction and establishes the first part of the theorem.

For the second part, since $R(f_k) = \widetilde{\Omega}(n^{2k+1})$ and $\mathrm{UC}(f_k) = \widetilde{O}(n^{k+1})$, we have $R(f_k) = \widetilde{\Omega}(\mathrm{UC}(f_k)^{2-\frac{1}{k+1}})$. Since we treated k as a constant, our notation hides constant and $\log n$ factors that depend only on k, i.e., we only get $R(f_k) \geq \left(\mathrm{UC}(f_k)^{2-\frac{1}{k+1}}\right) \Big/ \left(h_1(k)\log^{h_2(k)} n\right)$ for some functions $h_1(k)$ and $h_2(k)$. But we can always choose k to be a slow growing function of n so that these terms are negligible. This yields the desired separation $R(f) \geq \mathrm{UC}(f)^{2-o(1)}$. ◀

Clearly Theorem 1.2 and Theorem 1.3 follow from this, which we restate for convenience.

▶ **Theorem 1.2.** *There exists a total function f with $D(f) \geq \mathrm{UC}(f)^{2-o(1)}$.*

▶ **Theorem 1.3.** *There exists a total function f with $R(f) \geq \mathrm{UC}(f)^{2-o(1)}$.*

Deterministic communication vs. partition number

We now show Theorem 1.1 by lifting the previous separation to communication complexity.

From Theorem 1.3, we have a function $f : \{0,1\}^N \to \{0,1\}$ such that $R(f) \geq \mathrm{UC}(f)^{2-o(1)}$, which implies $D(f) \geq \mathrm{UC}(f)^{2-o(1)}$. Göös, Pitassi, and Watson [10] show that for any function f, there is a corresponding communication problem F such that

$$D^{\mathrm{cc}}(F) = \Omega(D(f)\log N) = \Omega(D(f)).$$

On the other hand, as explained in [10], we also have

$$\log \chi(F) = O(\mathrm{UC}(f)\log N) = \widetilde{O}(\mathrm{UC}(f)),$$

where we used the fact that our function has $N = n^{25^k}$, where k is a slow growing function of n, and hence $\log N = 25^k \log n = O(\log^2 n) = O(\log^2 \mathrm{UC}(f))$.

Since the conversion to communication complexity only weakens the result by log factors, the separation $D(f) \geq \mathrm{UC}(f)^{2-o(1)}$ immediately yields

$$D^{\mathrm{cc}}(F) \geq \left(\log \chi(F)\right)^{2-o(1)},$$

which establishes Theorem 1.1:

▶ **Theorem 1.1.** *There exists a function $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ with $D^{\mathrm{cc}}(F) \geq \left(\log \chi(F)\right)^{2-o(1)}$.*

6 Quantum query complexity vs. subcube partitions

In this section we establish Theorem 1.4 and Theorem 1.5. To show this we require a function Bkk from [3, Theorem 10] with the following properties.

▶ **Lemma 6.1.** *There exists a total function* $\textsc{Bkk} : \{0,1\}^{n^2} \to \{0,1\}$ *such that* $C(\textsc{Bkk}) = \widetilde{O}(n)$ *and* $Q(\textsc{Bkk}) = \widetilde{\Omega}(n^2)$.

We are now ready to prove Theorem 1.5, restated for convenience:

▶ **Theorem 1.5.** *There exists a total function* f *with* $Q(f) = \widetilde{\Omega}(\mathrm{UC}_1(f)^2)$.

Proof. Let $f = \textsc{Bkk}$. Then using Lemma 4.2 we know that $Q(f_{\mathrm{CS}}) = \Omega(Q(f)) = \widetilde{\Omega}(n^2)$ and $\mathrm{UC}_1(f_{\mathrm{CS}}) = O(C(f)\log^2 n) = \widetilde{O}(n)$. ◀

To show Theorem 1.4, we need another function BK-SUM, which is a variant of the K-SUM problem. It has the interesting property that any certificates for it consists essentially of input bits set to 0 and very few input bits set to 1. As shown in the proof of [3, Theorem 10], we have the following. (More precisely, our version of BK-SUM swaps the roles of zeros and ones compared to the function of [3], but this does not affect its quantum query complexity.)

▶ **Lemma 6.2.** *There exists a total function* $\textsc{bk-sum} : \{0,1\}^n \to \{0,1\}$ *with* $Q(\textsc{bk-sum}) = \widetilde{\Omega}(n)$ *such that for any function* f*, we have* $C(\textsc{bk-sum} \circ f) = O(nC_0(f) + C_1(f)\log^3 n)$.

In our construction, we repeatedly compose BK-SUM with other functions and hence we need to understand the behavior of BK-SUM under composition, analogous to Lemma 3.1 for AND and OR.

▶ **Lemma 6.3** (BK-SUM composition)**.** *For any function* f*, the following bounds hold:*

- $Q(\textsc{bk-sum}_n \circ f) = \widetilde{\Omega}(nQ(f))$
- $C(\textsc{bk-sum}_n \circ f) = O(nC_0(f) + C_1(f)\log^3 n)$.
- $\mathrm{UC}(\textsc{bk-sum}_n \circ f) \leq n\mathrm{UC}(f)$

Proof. The first lower bound follows because $Q(f \circ g) = \Theta(Q(f)Q(g))$ for any Boolean functions f and g [12, 26, 21] and we have $Q(\textsc{bk-sum}) = \widetilde{\Omega}(n)$ from Lemma 6.2. The second relation follows from Lemma 6.2. Lastly, $\mathrm{UC}(\textsc{bk-sum}_n \circ f) \leq n\mathrm{UC}(f)$ because this holds for any n-bit function. Any function $h \circ f$ can be unambiguously certified by showing all the outputs to f and providing unambiguous certificates for each output. ◀

We are now ready to establish the following theorem, which implies Theorem 1.4. This proof mimics the proof structure of Theorem 5.1 and reuses several ideas.

▶ **Theorem 6.4.** *For every* $k \geq 0$*, there exists a total Boolean function* $f_k : \{0,1\}^{N_k} \to \{0,1\}$*, such that* $Q(f_k) = \widetilde{\Omega}(n^{1.5k+1})$ *and* $\mathrm{UC}(f_k) = \widetilde{O}(n^{k+1})$*. Hence there is a function* f *with* $Q(f) \geq \mathrm{UC}(f)^{1.5-o(1)}$.

Proof. Let $f_1 = \textsc{And}_n \circ \textsc{Bkk}_{\mathrm{CS}}$, where BKK is the function on n^2 bits in Lemma 6.1. Let f_k be defined inductively as $f_k := \textsc{And}_n \circ (\textsc{bk-sum}_n \circ f_{k-1})_{\mathrm{CS}}$, i.e., f_k is the function obtain by composing $\textsc{And}_n$ with the cheat sheet version of $\textsc{bk-sum}_n$ composed with f_{k-1}.

We prove the claim by induction on k. The induction hypothesis and the base case, $f_1 = \textsc{And}_n \circ \textsc{Bkk}_{\mathrm{CS}}$, are presented below, where N_k is the input size of the function f_k.

Induction hypothesis (f_k)

- $N_k = O(n^{25^k})$
- $Q(f_k) = \widetilde{\Omega}(n^{1.5k+1})$
- $C_0(f_k) = \widetilde{O}(n^k)$
- $C_1(f_k) = \widetilde{O}(n^{k+1})$
- $\mathrm{UC}(f_k) = \widetilde{O}(n^{k+1})$

Base case ($f_1 = \textsc{And}_n \circ \textsc{Bkk}_{\mathrm{CS}}$)

- $N_1 = O(n^{25})$
- $Q(f_1) = \widetilde{\Omega}(n^{2.5})$
- $C_0(f_1) = \widetilde{O}(n)$
- $C_1(f_1) = \widetilde{O}(n^2)$
- $\mathrm{UC}(f_1) = \widetilde{O}(n^2)$

Let us first compute the complexities of $f_1 = \textsc{And}_n \circ \textsc{Bkk}_{\text{CS}}$ and verify that they are consistent with the induction hypothesis. First note that the input size of Bkk is n^2, and thus the input size of $\textsc{Bkk}_{\text{CS}}$ is $O(n^{24})$ (from Lemma 4.2), and hence the input size of $f_1 = \textsc{And}_n \circ \textsc{Bkk}_{\text{CS}}$ is $O(n^{25})$. We have $Q(f_1) = \widetilde{\Omega}(n^{2.5})$ since $Q(f_1) = \Omega(\sqrt{n}Q(\textsc{Bkk}_{\text{CS}})) = \widetilde{\Omega}(n^{2.5})$. The other inequalities follow straightforwardly from Lemma 3.1 and Lemma 6.1. We have $C_0(f_1) = \widetilde{O}(n)$ since $C_0(\textsc{And}_n \circ \textsc{Bkk}_{\text{CS}}) \leq C_0(\textsc{Bkk}_{\text{CS}}) = \widetilde{O}(n)$. We have $C_1(f_1) = \widetilde{O}(n^2)$ since $C_1(\textsc{And}_n \circ \textsc{Bkk}_{\text{CS}}) \leq nC_1(\textsc{Bkk}_{\text{CS}}) = \widetilde{O}(n^2)$. Lastly, we have $\text{UC}(f_1) = \text{UC}(\textsc{And}_n \circ \textsc{Bkk}_{\text{CS}}) \leq \text{UC}_0(\textsc{Bkk}_{\text{CS}}) + n\text{UC}_1(\textsc{Bkk}_{\text{CS}}) \leq D(\textsc{Bkk}_{\text{CS}}) + \widetilde{O}(nC(\textsc{Bkk})) = \widetilde{O}(D(\textsc{Bkk}) + nC(\textsc{Bkk})) = \widetilde{O}(n^2)$.

We now show that the induction hypothesis for f_k implies the same for f_{k+1}. The input size calculation is identical to that in Theorem 5.1 and hence we do not repeat it. The quantum query complexity of f_{k+1} can be lower bounded as follows:

$$\begin{aligned} Q(f_{k+1}) &= Q(\textsc{And}_n \circ (\textsc{bk-sum}_n \circ f_k)_{\text{CS}}) = \Omega(\sqrt{n}Q((\textsc{bk-sum}_n \circ f_k)_{\text{CS}})) \\ &= \Omega(\sqrt{n}Q(\textsc{bk-sum}_n \circ f_k)) = \Omega(n^{1.5}Q(f_k)) = \widetilde{\Omega}(n^{1.5(k+1)+1}), \end{aligned}$$

where we used Lemma 3.1, Lemma 4.2, Lemma 6.2, and Lemma 6.3 to compute the relevant measures.

Similarly we have

$$\begin{aligned} C_0(f_{k+1}) &= C_0(\textsc{And}_n \circ (\textsc{bk-sum}_n \circ f_k)_{\text{CS}}) \leq C_0((\textsc{bk-sum}_n \circ f_k)_{\text{CS}}) \\ &= \widetilde{O}(C(\textsc{bk-sum}_n \circ f_k)) = \widetilde{O}(nC_0(f_k) + C_1(f_k)) = \widetilde{O}(n^{k+1}) \qquad \text{and} \\ C_1(f_{k+1}) &= C_1(\textsc{And}_n \circ (\textsc{bk-sum}_n \circ f_k)_{\text{CS}}) \leq nC_1((\textsc{bk-sum}_n \circ f_k)_{\text{CS}}) \\ &= \widetilde{O}(nC(\textsc{bk-sum}_n \circ f_k)) = \widetilde{O}(n^2C_0(f_k) + nC_1(f_k)) = \widetilde{O}(n^{k+2}). \end{aligned}$$

Finally, using Lemma 3.1, Lemma 4.2, Lemma 6.2, and Lemma 6.3 again we have

$$\begin{aligned} \text{UC}(f_{k+1}) &= \text{UC}(\textsc{And}_n \circ (\textsc{bk-sum}_n \circ f_k)_{\text{CS}}) \\ &= O\left(\max\left\{\text{UC}_0((\textsc{bk-sum}_n \circ f_k)_{\text{CS}}), n\text{UC}_1((\textsc{bk-sum}_n \circ f_k)_{\text{CS}})\right\}\right) \\ &= \widetilde{O}\left(\max\left\{\text{UC}(\textsc{bk-sum}_n \circ f_k), nC(\textsc{bk-sum}_n \circ f_k)\right\}\right) \\ &= \widetilde{O}\left(\max\left\{n\text{UC}(f_k), n^2C_0(f_k) + nC_1(f_k)\right\}\right) = \widetilde{O}(n^{k+2}). \end{aligned}$$

This completes the induction and establishes the first part of the theorem. Using a similar argument in Theorem 5.1, this yields a function with $Q(f) \geq \text{UC}(f)^{1.5-o(1)}$. ◀

Finally, this establishes Theorem 1.4.

▶ **Theorem 1.4.** *There exists a total function f with $Q(f) \geq \text{UC}(f)^{1.5-o(1)}$.*

Acknowledgements. R. K. thanks Shalev Ben-David for several helpful conversations and comments on a preliminary draft.

This work is supported by ARO grant number W911NF-12-1-0486, the European Commission FET-Proactive project QALGO, ERC Advanced Grant MQC and Latvian State Research programme NexIT project No 1.

References

1 Scott Aaronson. Quantum certificate complexity. *SIAM Journal on Computing*, 35(4):804–824, 2006.

2 Scott Aaronson. Lower bounds for local search by quantum arguments. *Journal of Computer and System Sciences*, 74(3):313–332, 2008.

3 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th ACM Symposium on Theory of Computing (STOC 2016)*, to appear. arXiv:arXiv:1511.01937.

4 Alfred V. Aho, Jeffrey D. Ullman, and Mihalis Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the 15th ACM Symposium on Theory of Computing (STOC 1983)*, pages 133–139, 1983. doi:10.1145/800061.808742.

5 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing (special issue on quantum computing)*, 26:1510–1523, 1997.

6 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

7 Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some Boolean function complexity measures. In *Proceedings of 2013 IEEE Conference on Computational Complexity (CCC 2013)*, pages 185–196, June 2013. doi:10.1109/CCC.2013.27.

8 Mika Göös. Lower bounds for clique vs. independent set. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 1066–1076, 2015. doi:10.1109/FOCS.2015.69.

9 Mika Göös, T.S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. *Electronic Colloquium on Computational Complexity (ECCC) TR15-169*, 2015.

10 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 1077–1088, 2015. doi:10.1109/FOCS.2015.70.

11 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC 1996)*, pages 212–219, 1996. doi:10.1145/237814.237866.

12 Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC 2007)*, pages 526–535, 2007. doi:10.1145/1250790.1250867.

13 Rahul Jain and Hartmut Klauck. The partition bound for classical communication complexity and query complexity. In *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*, CCC'10, pages 247–258, 2010. doi:10.1109/CCC.2010.31.

14 Rahul Jain, Troy Lee, and Nisheeth K. Vishnoi. A quadratically tight partition bound for classical communication complexity and query complexity. *arXiv preprint* arXiv:1401.4512, 2014.

15 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*. Algorithms and Combinatorics. Springer, 2012.

16 Jedrzej Kaniewski, Troy Lee, and Ronald de Wolf. Query complexity in expectation. In *Automata, Languages, and Programming*, volume 9134 of *Lecture Notes in Computer Science*, pages 761–772. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-47672-7_62.

17 Robin Kothari, David Racicot-Desloges, and Miklos Santha. Separating Decision Tree Complexity from Subcube Partition Complexity. In *Approximation, Randomization, and*

Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015), volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 915–930, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.915.

18 Eyal Kushilevitz, Nathan Linial, and Rafail Ostrovsky. The linear-array conjecture in communication complexity is false. *Combinatorica*, 19(2):241–254, 1999. doi:10.1007/s004930050054.

19 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 2006.

20 Sophie Laplante and Frédéric Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. *SIAM Journal on Computing*, 38(1):46–62, 2008.

21 Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 344–353, 2011. arXiv:arXiv:1011.3020, doi:10.1109/FOCS.2011.75.

22 Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends® in Theoretical Computer Science*, 3(4):263–399, 2007. doi:10.1561/0400000040.

23 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), July 2014.

24 Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. doi:10.1137/0220062.

25 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 15(4):557–565, 1995.

26 Ben W. Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, SODA'11, pages 560–569, 2011. URL: http://dl.acm.org/citation.cfm?id=2133036.2133080.

27 Petr Savický. On determinism versus unambiquous nondeterminism for decision trees. *ECCC*, TR02-009, 2002.

28 Robert Špalek and Mario Szegedy. All quantum adversary methods are equivalent. *Theory of Computing*, 2(1):1–18, 2006.

29 Avishay Tal. Properties and applications of Boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS'13, pages 441–454, 2013. doi:10.1145/2422436.2422485.

30 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th ACM Symposium on Theory of Computing*, STOC'79, pages 209–213, 1979. doi:10.1145/800135.804414.