# Separations in query complexity using cheat sheets

Scott Aaronson
MIT
aaronson@csail.mit.edu

Shalev Ben-David
MIT
shalev@mit.edu

Robin Kothari
MIT
rkothari@mit.edu

**Abstract**

We show a power 2.5 separation between bounded-error randomized and quantum query complexity for a total Boolean function, refuting the widely believed conjecture that the best such separation could only be quadratic (from Grover's algorithm). We also present a total function with a power 4 separation between quantum query complexity and approximate polynomial degree, showing severe limitations on the power of the polynomial method. Finally, we exhibit a total function with a quadratic gap between quantum query complexity and certificate complexity, which is optimal (up to log factors). These separations are shown using a new, general technique that we call the cheat sheet technique. The technique is based on a generic transformation that converts any (possibly partial) function into a new total function with desirable properties for showing separations. The framework also allows many known separations, including some recent breakthrough results of Ambainis et al., to be shown in a unified manner.

## Contents

# 1 Introduction

Query complexity (or decision tree complexity) is a model of computation that allows us to examine the strengths and weaknesses of resources such as access to randomness, nondeterminism, quantum computation and more. As opposed to the Turing machine model where we can only conjecture that certain resources exponentially speed up computation, in this model such beliefs can be proved. In particular, the query model is a natural setting to describe several well-known quantum algorithms, such as Grover's algorithm [Gro96] and quantum walks [Amb07]; even Shor's algorithm for integer factorization [Sho97] is based on a quantum query algorithm.

In the query model we measure a problem's complexity, called its query complexity, by the minimum number of input bits that have to be read to solve the problem on any input. For a function $f$, we use $D(f)$, $R(f)$, and $Q(f)$ to denote the query complexity of computing $f$ with a deterministic, bounded-error randomized, and bounded-error quantum algorithm respectively.

For a total Boolean function $f : \{0,1\}^n \to \{0,1\}$, we know that these measures are polynomially related: in one direction, we have $Q(f) \leq R(f) \leq D(f)$ as each model is more powerful than the next, and in the other direction, we have $D(f) = O(R(f)^3)$ [Nis91] and $D(f) = O(Q(f))^6$ [BBC+01].

Until recently, the best known separation between $D(f)$ and $R(f)$ was the NAND-tree function [SW86], which satisfies $D(f) = \Omega(R(f)^{1.3267})$ and even $D(f) = \Omega(R_0(f)^{1.3267})$, where $R_0(f)$ denotes zero-error randomized query complexity. The best known separation between $D(f)$ or $R(f)$ versus $Q(f)$ was only quadratic, achieved by the OR function: $D(\text{OR}) \geq R(\text{OR}) = \Omega(n)$ and $Q(\text{OR}) = \Theta(\sqrt{n})$ [Gro96, BBBV97]. No separation was known between $R_0(f)$ and $R(f)$. Furthermore, it was believed that all of these separations were optimal.

These beliefs were shattered by a recent breakthrough by Ambainis, Balodis, Belovs, Lee, Santha, and Smotrovs [ABB+15], who built upon the techniques of Göös, Pitassi, and Watson [GPW15], and showed the near-optimal separations $D(f) = \widetilde{\Omega}(R_0(f)^2)$ and $R_0(f) = \widetilde{\Omega}(R(f)^2)$. Additionally, they presented new separations between quantum and classical models, exhibiting functions with $D(f) = \widetilde{\Omega}(Q(f)^4)$ and $R_0(f) = \widetilde{\Omega}(Q(f)^3)$. However, no super-quadratic separation was shown between a quantum and classical model when they are both allowed to err in the same way (e.g., bounded error, zero error, etc.) For example, no super-quadratic separation was shown between $Q(f)$ and its classical analogue, $R(f)$.

## 1.1 Results

### Randomized versus quantum query complexity

We present the first super-quadratic separation between quantum and classical query complexity when both models are allowed bounded error. This is a counterexample to a conjecture that had been widely believed about the nature of quantum speed-ups: that if the function is total, so that the inputs are not handpicked to orchestrate a quantum speed-up, then the best quantum advantage possible is the quadratic speed-up achieved by Grover's algorithm.

**Theorem 1.** *There exists a total function $f$ such that $R(f) = \widetilde{\Omega}(Q(f)^{2.5})$.*

Theorem 1 can be boosted to a cubic separation, i.e., $R(f) = \widetilde{\Omega}(Q(f)^3)$, in a completely black-box manner if there exists a partial function (a function defined only on a subset of $\{0,1\}^n$) with $\widetilde{\Omega}(n)$ randomized query complexity but only $\text{poly}(\log(n))$ quantum query complexity (the best possible separation between these measures up to log factors). It is conjectured that a recently studied partial function called $k$-fold Forrelation achieves this separation [AA15].

### Quantum query complexity versus polynomial degree

Nisan and Szegedy [NS95] introduced two new measures of a Boolean function $f$ called the degree and approximate degree, denoted $\deg(f)$ and $\widetilde{\deg}(f)$ respectively. The (approximate) degree of a function $f$ is the minimum degree of a polynomial over the input variables that (approximately) equals $f(x)$ at every input $x \in \{0,1\}^n$. They introduced these measures to lower bound classical query complexity, showing that $\deg(f) \leq D(f)$ and $\widetilde{\deg}(f) \leq R(f)$. It turns out that these measures also lower bound the corresponding quantum measures, since $\deg(f) \leq 2\,Q_E(f)$ and $\widetilde{\deg}(f) \leq 2\,Q(f)$ [BBC+01], where $Q_E(f)$ denotes the exact quantum query complexity of $f$.

Approximate degree has proved to be a fruitful lower bound technique for quantum query complexity, especially for problems like the collision and element distinctness problems [AS04], where it is known that the original quantum adversary method of Ambainis, another commonly used lower bound technique, cannot show nontrivial lower bounds.

For any lower bound technique, it is natural to ask whether there exist functions where the technique fails to prove a tight lower bound. Answering this question, Ambainis showed that the approximate degree of a function can be asymptotically smaller than its quantum query complexity by exhibiting a function with $Q(f) = \Omega(\widetilde{\deg}(f)^{1.3219})$ [Amb03]. We dramatically strengthen this separation to obtain nearly a $4^{\text{th}}$ power gap.

**Theorem 2.** *There exists a total function $f$ such that $Q(f) \geq \widetilde{\deg}(f)^{4-o(1)}$.*

Theorem 2 is optimal assuming the conjecture that $D(f) = O(\mathrm{bs}(f)^2)$, where $\mathrm{bs}(f)$ is block sensitivity (to be defined in Section 2.2).

### The cheat sheet technique

These separations are shown using a new technique for proving separations between query measures, which we call the cheat sheet technique. Our technique is based on a generic transformation that takes any (partial or total) function and transforms it into a "cheat sheet version" of the function that has desirable properties for proving separations.

While the strategy is inspired by the recent breakthrough results [GPW15, ABB+15] (and bears some similarity to older works [LG06, AdW14]), it represents a more general approach: it provides a framework for proving separations and allows many separations to be shown in a unified manner. Thus the task of proving separations is reduced to the simpler task of finding the right function to plug into this construction. For example, it can be used to convert a partial function separation between two models into a weaker total function separation.

In this paper we demonstrate the power of the cheat sheet technique by using it to exhibit several new total function separations in query complexity.

### Other separations

On the path to proving Theorem 2, we show several new separations. First, we quadratically separate quantum query complexity from certificate complexity, which is essentially optimal since $Q(f) \leq C(f)^2$.

**Theorem 3.** *There exists a total function $f$ such that $Q(f) = \widetilde{\Omega}(C(f)^2)$.*

Besides being a stepping stone to proving Theorem 2, the question of $Q(f)$ versus $C(f)$ has been studied because it is known that the original adversary method of Ambainis (also known as the positive-weights adversary method) cannot prove a lower bound greater than $C(f)$ [ŠS06, HLŠ07].

Plugging the function of Theorem 3 into the cheat sheet framework directly yields a function whose quantum query complexity is quadratically larger than its (exact) degree, improving the recent result of [GPW15], who exhibited a function with $D(f) = \widetilde{\Omega}(\deg(f)^2)$.

**Theorem 4.** *There exists a total function $f$ such that $Q(f) = \widetilde{\Omega}(\deg(f)^2)$.*

This theorem works in a very black-box way: any separation between a measure like $Q(f)$ or $R(f)$ and $C(f)$ can be plugged into the cheat sheet technique to obtain the same separation (up to log factors) between that measure and exact degree. For example, since the AND-OR function satisfies $R(f) = \Omega(C(f)^2)$, the cheat sheet version of AND-OR satisfies $R(f) = \widetilde{\Omega}(\deg(f)^2)$.

This result also shows limitations on using $\deg(f)$ to lower bound $Q_E(f)$, since $\deg(f)$ can sometimes be quadratically smaller than $Q_E(f)$ and can even be quadratically smaller than $Q(f)$.

### Summary of results

We summarize our new results in the table below.

| Separation achieved | Known relation | Result |
|---|---|---|
| $R(f) = \widetilde{\Omega}(Q(f)^{2.5})$ | $R(f) = O(Q(f)^6)$ | Theorem 1 |
| $Q(f) \geq \widetilde{\deg}(f)^{4-o(1)}$ | $Q(f) = O(\widetilde{\deg}(f)^6)$ | Theorem 2 |
| $Q(f) = \widetilde{\Omega}(C(f)^2)$ | $Q(f) = O(C(f)^2)$ | Theorem 3 |
| $Q(f) = \widetilde{\Omega}(\deg(f)^2)$ | $Q(f) = O(\deg(f)^3)$ | Theorem 4 |

Table 1: New separations shown in this paper

We are also able to use the cheat sheet technique to reprove many of the query separations of [GPW15, ABB+15, GJPW15]. Some of their results are subsumed by the results in Table 1. We also prove $R_0(f) = \widetilde{\Omega}(Q(f)^3)$ (Theorem 23) and $R(f) = \widetilde{\Omega}(Q_E(f)^{3/2})$ (Theorem 24) in Section 6.4. For more details, see Table 2 in which we summarize our results and the best known separations and relations between query measures.

It is also worth pointing out what we are unable to reproduce with our framework. The only separations from these papers that we do not reproduce are those that make essential use of "back pointers". Reproducing these separations in the cheat sheet framework is an interesting direction for future research.

## 1.2 Overview of techniques

### Cheat sheet technique and randomized versus quantum query complexity

While we know large partial function separations between randomized and quantum query complexity, such as Simon's problem [Sim97] or the Forrelation problem [AA15] that satisfies $R(g) = \widetilde{\Omega}(\sqrt{n})$ and $Q(g) = 1$, it is unclear how to make these functions total. We could simply define $g$ to be 0 on inputs not in the domain, but then the quantum algorithm would need to be able to decide whether an input is in the domain.

To solve this problem, we compose the Forrelation problem $g$ with a total function $h = \text{AND-OR}$ on $n^2$ bits, to obtain a partial function $f = g \circ h$ on $n^3$ bits that inherits the properties of both $g$ and $h$. Since AND-OR has low certificate complexity, it is easier to certify that an input lies in the domain of $g \circ h$, since we only need to certify the outputs of the $n$ different $h$ gates.

Table 2: Best known separations between complexity measures

| | $D$ | $R_0$ | $R$ | $C$ | RC | bs | $Q_E$ | deg | $Q$ | $\widetilde{\deg}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $D$ | | 2, 2<br>[ABB+15] | 2*, 3<br>[ABB+15] | 2, 2<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2, 3<br>[ABB+15] | 2, 3<br>[GPW15] | 4*, 6<br>[ABB+15] | 4*, 6<br>[ABB+15] |
| $R_0$ | 1, 1<br>$\oplus$ | | 2, 2<br>[ABB+15] | 2, 2<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2, 3<br>[ABB+15] | 2, 3<br>[GJPW15] | 3, 6<br>[ABB+15] | 4*, 6<br>[ABB+15] |
| $R$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | | 2, 2<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 1.5, 3<br>[ABB+15] | 2, 3<br>[GJPW15] | 2.5, 6<br>Th. 1 | 4*, 6<br>[ABB+15] |
| $C$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 2<br>$\oplus$ | | 2, 2<br>[GSS13] | 2, 2<br>[GSS13] | 1.1527, 3<br>[Amb13] | $\log_3 6$, 3<br>[NW95] | 2, 4<br>$\wedge$ | 2, 4<br>$\wedge$ |
| RC | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | | 1.5, 2<br>[GSS13] | 1.1527, 2<br>[Amb13] | $\log_3 6$, 2<br>[NW95] | 2, 2<br>$\wedge$ | 2, 2<br>$\wedge$ |
| bs | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | | 1.1527, 2<br>[Amb13] | $\log_3 6$, 2<br>[NW95] | 2, 2<br>$\wedge$ | 2, 2<br>$\wedge$ |
| $Q_E$ | 1, 1<br>$\oplus$ | 1.3267, 2<br>$\bar{\wedge}$-tree | 1.3267, 3<br>$\bar{\wedge}$-tree | 2, 2<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | | 2, 3<br>Th. 4 | 2, 6<br>$\wedge$ | 4*, 6<br>Th. 2 |
| deg | 1, 1<br>$\oplus$ | 1.3267, 2<br>$\bar{\wedge}$-tree | 1.3267, 3<br>$\bar{\wedge}$-tree | 2, 2<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 2*, 3<br>$\wedge \circ \vee$ | 1, 1<br>$\oplus$ | | 2, 6<br>$\wedge$ | 2, 6<br>$\wedge$ |
| $Q$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 2, 2<br>Th. 3 | 2*, 3<br>Th. 3 | 2*, 3<br>Th. 3 | 1, 1<br>$\oplus$ | 2, 3<br>Th. 4 | | 4*, 6<br>Th. 2 |
| $\widetilde{\deg}$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 7/6, 2<br>$\wedge \circ$ ED | 7/6, 3<br>$\wedge \circ$ ED | 7/6, 3<br>$\wedge \circ$ ED | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | 1, 1<br>$\oplus$ | |

An entry $a, b$ in the row $M_1$ and column $M_2$ roughly[a] means $M_1(f) = \widetilde{O}(M_2(f)^b)$ for all total $f$ and there exists a total $f$ with $M_1(f) = \widetilde{\Omega}(M_2(f)^a)$. Each cell contains a citation or a description of a separating function, where $\oplus$ = PARITY, $\wedge$ = AND, $\wedge \circ \vee$ = AND-OR, $\bar{\wedge}$-tree is the balanced NAND-tree function [SW86, San95], and $\wedge \circ$ ED is the AND function composed with Element Distinctness.[b] Entries followed by a star (e.g., 2*) correspond to separations that are optimal if $D(f) = O(\mathrm{bs}(f)^2)$. Separations colored red are new to this work. Separations colored gray are separations from recent papers that we reprove in this work.

[a] More precisely it means $a \leq \mathrm{crit}(M_1, M_2) \leq b$, where we define $\mathrm{crit}(\cdot, \cdot)$ in Section 2.

[b] The Element Distinctness function accepts a list of $n$ numbers in $[n]$ as input and asks if any two of them are equal. The certificate complexity of the $\mathrm{AND}_n$ composed with element distinctness on $n$ bits is $\widetilde{O}(n)$, by composing certificates for the two functions. However, its approximate degree is $\widetilde{\Omega}(n^{7/6})$, which follows from noting that its (negative) one-sided approximate degree is $\Omega(n^{2/3})$ [BT15] and that composition with the $\mathrm{AND}_n$ function raises its approximate degree by a factor of $\sqrt{n}$ [BT15, She13]. (We thank Mark Bun for outlining this proof.)

Since $R(\text{AND-OR}_{n^2}) = \Omega(n^2)$, it can be shown that $R(f) = R(g \circ h) = \Omega(n^{2.5})$. However, since $Q(\text{AND-OR}_{n^2}) = O(n)$, we have $Q(f) = O(n)$. Now $f$ is still a partial function, but there is a small certificate for the input being in the domain. If we could ensure that only the quantum algorithm had access to this certificate, but the randomized algorithm did not, we would have our power 2.5 separation.

To achieve this, we hide a "cheat sheet" in the input that contains all the information the quantum algorithm needs. We store it in a vast array of size $n^{10}$ of potential cheat sheets, which cannot be searched quickly by brute force by any algorithm, classical or quantum. Thus, we must provide the quantum algorithm the address of the correct cheat sheet. But what information do we have that is known only to the quantum algorithm? The value of $f$ on the input! While one bit does not help much, we can use $10 \log n$ copies of $f$ acting on $10 \log n$ different inputs. The outputs to these $10 \log n$ problems can index into an array of size $n^{10}$, and can be determined by the quantum algorithm using only $O(n \log n)$ queries. At this index we will store the cheat sheet, which contains certificates for all $10 \log n$ instances of $f$, convincing us that all the inputs lie in the domain. Our cheat sheet function now evaluates to 1 if and only if all $10 \log n$ instances of $f$ lie in the domain, *and* the cell in the array pointed to by the outputs of these $10 \log n$ instances of $f$ contains a cheat sheet certifying that these inputs lie in the domain.

### Quantum query complexity versus certificate complexity

Consider the $k$-SUM problem, in which we are given $n$ numbers and have to decide if any $k$ of them sum to 0 (mod $M$) for some $M$. For large $k$, the quantum query complexity of $k$-SUM is nearly linear in $n$ [BŠ13]. While it is easy to certify 1-inputs by showing any $k$ elements that sum to 0, the 0-inputs are difficult to certify and hence the certificate complexity is also linear in $n$.

Building on $k$-SUM, we define a new function that we call BLOCK $k$-SUM, whose quantum query complexity and certificate complexity are both linear in $n$. However, BLOCK $k$-SUM has a curious property: for both 1-inputs and 0-inputs, the certificates themselves consist almost exclusively of input bits set to 1. This means that if we compose this function with $k$-SUM, the composed function on $n^2$ bits has certificates of size $\widetilde{O}(n)$ because any certificate of BLOCK $k$-SUM consists almost entirely of 1s, which correspond to $k$-SUM instances that output 1, which are easy to certify. On the other hand, the quantum query complexity of the composed function, which we call BKK for BLOCK $k$-SUM of $k$-SUM, is the product of the quantum query complexities of individual functions. Thus the certificate complexity of BKK is $\widetilde{O}(n)$, but its quantum query complexity is $\widetilde{\Omega}(n^2)$.

### Quantum query complexity versus polynomial degree

We show that plugging any function that achieves $Q(f) = \widetilde{\Omega}(n^2)$ and $C(f) = \widetilde{O}(n)$ into the cheat sheet framework yields a function with $Q(f) = \widetilde{\Omega}(n^2)$ and $\deg(f) = \widetilde{O}(n)$. The quantum lower bound uses the hybrid method [BBBV97] and the recent strong direct product theorem for quantum query complexity [LR13]. The degree upper bound holds because for every potential cheat sheet location, there exists a degree $\widetilde{O}(n)$ polynomial that checks if this location is the one pointed to by the given input. And since the input can point to at most one cheat sheet, the sum of these polynomials equals the function $f$.

To achieve a fourth power separation between $Q(f)$ and $\widetilde{\deg}(f)$, we need to check whether a cheat sheet is valid using a polynomial of degree $\widetilde{O}(\sqrt{C(f)})$. Some certificates can be checked by Grover's algorithm in time $\widetilde{O}(\sqrt{C(f)})$, which would yield an approximating polynomial of similar degree, but this is not true for all functions $f$. To remedy this, we construct a new function based on composing BKK with itself recursively $\log n$ times, to obtain a new function we call RECBKK,

and show that certificates for RecBKK can be checked quickly by a quantum algorithm.

## 2 Preliminaries

For any positive integer $n$, let $[n] := \{1, \ldots, n\}$. We use $f(n) = \widetilde{O}(g(n))$ to mean there exists a constant $k$ such that $f(n) = O(g(n)\log^k n)$. For example, $f(n) = \widetilde{O}(1)$ means $f(n) = O(\log^k n)$ for some constant $k$. Similarly, $f(n) = \widetilde{\Omega}(g(n))$ denotes $f(n) = \Omega(g(n)/\log^k n)$ for some constant $k$. Finally, $f(n) = \widetilde{\Theta}(g(n))$ means $f(n) = \widetilde{O}(g(n))$ and $f(n) = \widetilde{\Omega}(g(n))$.

### 2.1 Boolean functions

Let $f$ be function from a domain $D \subseteq \Sigma^n$ to $\{0,1\}$, where $\Sigma$ is some finite set. Usually $\Sigma = \{0,1\}$, and we assume this for simplicity below, although most complexity measures are easily generalized to larger input alphabets. A function $f$ is called a total function if $D = \Sigma^n$. Otherwise, we say $f$ is a partial function or a promise problem and refer to $D$ as the promised set or the promise.

For any two (possibly partial) Boolean functions $g : G \to \{0,1\}$, where $G \subseteq \{0,1\}^n$, and $h : H \to \{0,1\}$, where $H \subseteq \{0,1\}^m$, we can define the composed function $g \circ h : D \to \{0,1\}$, where $D \subseteq \{0,1\}^{nm}$, as follows. For an input $(z_1, z_2, \ldots, z_{nm}) \in \{0,1\}^{nm}$, we define $g \circ h$ as follows (as depicted in Figure 1):

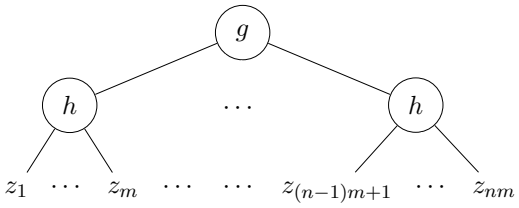

Figure 1: Composed function $g \circ h$.

$$g \circ h(z) := g(h(z_1, \ldots, z_m), h(z_{m+1}, \ldots, z_{2m}), \ldots, h(z_{(n-1)m+1}, \ldots, z_{nm})), \tag{1}$$

where the function $g \circ h$ is only defined on $z$ where the inputs to the $h$ gates lie in the domain $H$ and the $n$-bit input to $g$, $(h(z_1, \ldots, z_m), h(z_{m+1}, \ldots, z_{2m}), \ldots, h(z_{(n-1)m+1}, \ldots, z_{nm}))$, lies in $G$.

Some Boolean functions that we use in this paper are $\textsc{And}_n$ and $\textsc{Or}_n$, the And and Or functions respectively on $n$ bits, and $\textsc{And-Or}_{n^2} : \{0,1\}^{n^2} \to \{0,1\}$, defined as $\textsc{And}_n \circ \textsc{Or}_n$.

### 2.2 Complexity measures

Formal definitions of most measures introduced here may be found in the survey on query complexity by Buhrman and de Wolf [BdW02] and in the recent paper of Ambainis et al. [ABB+15].

In the model of query complexity, algorithms may only access the input by asking queries to an oracle that knows the input $x \in D$. The queries are of the form "what is $x_i$?" for some $i \in [n]$ and the oracle responds with the value of $x_i \in \Sigma$. The goal is to minimize the number of queries made to compute $f$.

We use $D(f)$ to denote the deterministic query complexity of computing $f$, the minimum number of queries that have to be made by a deterministic algorithm that outputs $f(x)$ on every input $x$. We use $R(f)$ to denote bounded-error randomized query complexity, the minimum number of queries made by a randomized algorithm that outputs $f(x)$ on input $x$ with probability at least $2/3$. We use $R_0(f)$ to denote zero-error randomized query complexity, the minimum number of queries made by a randomized algorithm that must output either the correct answer, $f(x)$, on input $x$ or output $*$, indicating that it does not know the answer. However, the probability (over the internal randomness of the algorithm) that it outputs $*$ should be at most $1/2$ on any $x \in D$.

Similarly, we study quantum analogues of these measures. The quantum analogue of $D(f)$ is exact quantum query complexity, denoted $Q_E(f)$, the minimum number of queries made by a quantum algorithm that outputs $f(x)$ on every input $x \in D$ with probability 1. The quantum

analogue of $R(f)$ is bounded-error quantum query complexity, denoted $Q(f)$, where the algorithm is only required to be correct with probability at least $2/3$.

We also study some measures related to the complexity of certifying the output. For a function $f$, let $C_1(f)$ denote the minimum number of bits of a 1-input, i.e., an input with $f(x) = 1$, that have to be revealed to convince a verifier that $f(x) = 1$. Alternately, imagine the verifier is allowed to interact with an untrustworthy prover who claims that $f(x) = 1$, and is allowed to provide an arbitrarily long witness for this. After receiving the witness, the verifier makes at most $C_1(f)$ queries to the input and decides whether to accept that $f(x) = 1$ or not. We require that if $f(x) = 1$, then there is some witness that makes the verifier accept and if $f(x) = 0$ then no witness makes the verifier accept. Similarly we define $C_0(f)$ as the analogous notion for 0-inputs. Finally the certificate complexity of $f$, or $C(f)$, is defined as $\max\{C_0(f), C_1(f)\}$. Similarly, if the verifier is randomized and only needs to accept valid proofs and reject invalid proofs with probability greater than $2/3$, we get the bounded-error randomized analogues $\mathrm{RC}_0(f)$, $\mathrm{RC}_1(f)$, and $\mathrm{RC}(f) := \max\{\mathrm{RC}_0(f), \mathrm{RC}_1(f)\}$ [Aar06].

We also study a combinatorial measure of a function called block sensitivity, denoted $\mathrm{bs}(f)$. For a function $f$ and input $x$, a block of bits $B \subseteq [n]$ is said to be sensitive for $x$ if the input obtained by complementing all the bits $x_i$ for $i \in B$ yields an input $y \in D$ with $f(x) \neq f(y)$. The block sensitivity of $f$ on input $x$ is the maximum number of disjoint blocks sensitive for $x$. The block sensitivity of $f$ is the maximum block sensitivity over all inputs $x \in D$.

Lastly we study measures related to representing a function with a polynomial with real coefficients. The (exact) degree of $f$, denoted $\deg(f)$, is the minimum degree of a real polynomial $p$ over the variables $x_1, x_2, \ldots, x_n$, such that $p(x_1, x_2, \ldots, x_n) = f(x)$ for all $x \in D$. Similarly, the approximate degree of $f$, denoted $\widetilde{\deg}(f)$, is the minimum degree of a polynomial that approximates the value of $f(x)$ on every input $x \in D$, i.e., for all $x \in D$ the polynomial satisfies $|p(x) - f(x)| \leq 1/3$.

All the measures introduced in this section lie between 0 and $n$. For most measures this follows because the algorithm may simply query all $n$ input variables and compute $f(x)$. Also, any Boolean function $f$ can be exactly represented by a real polynomial of degree at most $n$.

### 2.3 Known relations and separations

We now summarize the best known relations between the complexity measures studied in this paper. Figure 2 depicts all known relations of the type $M_1(f) = O(M_2(f))$ for complexity measures $M_1(f)$ and $M_2(f)$. An upward line from $M_1$ to $M_2$ in Figure 2 indicates $M_1(f) = O(M_2(f))$ for all (partial or total) Boolean functions $f$. Most of these relations follow straightforwardly from definitions. For example $R_0(f) \leq D(f)$ because any deterministic algorithm is also a zero-error randomized algorithm. The relationships that do not follow from such observations are $\mathrm{bs}(f) = O(\mathrm{RC}(f))$ [Aar06], $\deg(f) = O(Q_E(f))$ [BBC+01], and $\widetilde{\deg}(f) = O(Q(f))$ [BBC+01].

All other known relationships between these measures follow by combining the relationships in Figure 2 with the following relationships that hold for all total Boolean functions:

- $C(f) \leq \mathrm{bs}(f)^2$ [Nis91]
- $D(f) \leq C(f)\,\mathrm{bs}(f)$ [BBC+01]
- $D(f) \leq \deg(f)^3$ [Mid04, Tal13]
- $\mathrm{RC}(f) = O(\widetilde{\deg}(f)^2)$ [KT13]
- $R_0(f) = O(R(f)^2 \log R(f))$ [KT13]

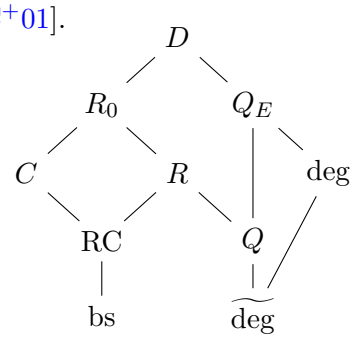


Figure 2: Relations between complexity measures.

We now turn to the best known separations between these measures. To conveniently express these results, we use a notion called

the *critical exponent*, as defined by [GSS13]. The the critical exponent for a measure $M_1$ relative to a measure $M_2$, denoted $\mathrm{crit}(M_1, M_2)$, is the infimum over all $r$ such that the relation $M_1(f) = O(M_2(f)^r)$ holds for all total functions $f$. For example, because $D(f) \leq C(f)^2$ for all total $f$, we have $\mathrm{crit}(D, C) \leq 2$. Furthermore, since we also know that the AND-OR function on $k^2$ bits satisfies $D(f) = k^2$ and $C(f) = k$, we have $\mathrm{crit}(D, C) = 2$. Note that the critical exponent between $M_1$ and $M_2$ is 2 even if we have $M_1(f) = O(M_2(f)^2 \log M_2(f))$, or more generally if $M_1(f) \leq M_2(f)^{2+o(1)}$.

Table 2 lists the best known separations between these measures. A cell in the table has the form $a, b$, where $a \leq b$ are the best known lower and upper bounds on the critical exponent for the row measure relative to the column measure. The cell also contains a citation for the function that provides the lower bound. For example, in the cell corresponding to row $D$ and column deg, we have the entry $2, 3$, which means $2 \leq \mathrm{crit}(D, \mathrm{deg}) \leq 3$, and a function achieving the separation appears in [GPW15].

# 3 Randomized versus quantum query complexity

In this section we show a power 2.5 separation between bounded-error randomized query complexity, $R(f)$, and bounded-error quantum query complexity, $Q(f)$. In Section 3.1 we motivate the cheat sheet framework and provide a sketch of the separation. In Section 3.2 we formally prove the separation.

## 3.1 Intuition

We begin with the best known separation between randomized and quantum query complexity for *partial* functions, which is currently $\widetilde{\Omega}(\sqrt{n})$ versus 1 provided by the Forrelation problem [AA15]. (Note that Simon's problem also provides a similar separation up to log factors [Sim97].) Let $g$ be the Forrelation function on $D \subseteq \{0,1\}^n$ with $R(g) = \widetilde{\Omega}(\sqrt{n})$ and $Q(g) = 1$, although any function with these query complexities (up to log factors) would do.

One way to make $g$ total would be to define it to be 0 on the rest of the domain $\{0,1\}^n \setminus D$. But then it is unclear if the new function $g$ has low quantum query complexity, since the quantum algorithm would have to test whether the input lies in the promised set $D$. In general since partial function separations often require a stringent promise on the input, we may expect that it is necessary to examine most input bits to ascertain that $x \in D$.

Indeed, it is not even clear how to *certify* that $x \in D$ for our function $g$. On the other hand, the domain certification problem is trivial for total functions. So we can compose $g$ with a total function $h$ to obtain some of the desirable properties of both. We can certify that an input to $g \circ h$ lies in its domain by certifying the outputs of all instances of $h$. This means we want $h$ to have low certificate complexity, and since we will use $g \circ h$ to separate randomized and quantum query complexity, we would also like $h$ to have $Q(h)$ smaller than $R(h)$. A function that fits the bill perfectly is the $\text{AND-OR}_{m^2}$ function that satisfies $R(h) = \Omega(m^2)$, $C(h) = m$, and $Q(h) = O(m)$.

We can now compute the various complexities of $f := g \circ h$. First we have $Q(f) = \widetilde{O}(m)$, by composing algorithms for $g$ and $h$. There also exists a certificate of size $\widetilde{O}(nm)$ that proves that the input satisfies the promise, since we can certify the outputs of the $n$ different $h$ gates using certificates of size $C(h) = m$. Also, given query access to this certificate of size $\widetilde{O}(nm)$, a quantum algorithm can check the certificate's validity using Grover search in $\widetilde{O}(\sqrt{nm})$ queries. Hence a quantum algorithm with access to a certificate can solve the problem with $\widetilde{O}(m + \sqrt{nm})$ queries. On the other hand, it can be shown that $R(f) = R(g \circ h) = \Omega(R(g)R(h)) = \widetilde{\Omega}(\sqrt{n}m^2)$.

Now if we set $m = n$, then we see that $Q(f) = \widetilde{O}(n)$ and $R(f) = \widetilde{\Omega}(n^{2.5})$, but $f$ is still a partial function. However, $f$ has the desirable property that a quantum algorithm given query access to a certificate can decide whether the input satisfies the promise using $\widetilde{O}(n)$ queries. But we cannot simply append the certificate to the input as we do not want the randomized algorithm to be able to use it. It is, however, acceptable if the randomized algorithm finds the certificate after it spends $R(f)$ queries, since that is the lower bound we want to prove.

What we would like is to hide the certificate somewhere in the input where only the quantum algorithm can find it. What information do we have that is only known to the quantum algorithm and remains unknown to the randomized algorithm unless it spends $R(f)$ queries? Clearly the value of $f$ on the input has this property.

While one bit does not help much, we can obtain additional bits of this kind by using (say) $10 \log n$ copies of $f$. Now the answer string to these problems can address an array of size $n^{10}$, much larger than can be brute-force searched by a randomized algorithm in a reasonable amount of time. Furthermore, this address can be found by a quantum algorithm using only $\widetilde{O}(Q(f))$ queries. At this location we will store a cheat sheet: a collection of certificates for all $10 \log n$ instances of $f$.

Now the new function, which we call $f_{\mathrm{CS}}$ (shown in Figure 3), evaluates to 1 if and only if the $10 \log n$ inputs are in the domain of $f$, and the outputs of the $10 \log n$ copies of $f$ point to a valid cheat sheet, i.e., a set of valid certificates for the $10 \log n$ copies of $f$. This construction ensures that the quantum algorithm can find the cheat sheet using $\widetilde{O}(Q(f)) = \widetilde{O}(n)$ queries, and then verify it using $\widetilde{O}(n)$ queries, which gives $Q(f_{\mathrm{CS}}) = \widetilde{O}(n)$. On the other hand, we intuitively expect that the randomized algorithm cannot even find the cheat sheet unless it computes $f$ by spending at least $R(f) = \widetilde{\Omega}(n^{2.5})$ queries, which gives us the desired separation.

### 3.2 Implementation

In this section we prove Theorem 1, restated for convenience:

**Theorem 1.** *There exists a total function $f$ such that $R(f) = \widetilde{\Omega}(Q(f)^{2.5})$.*

Let $g$ be the Forrelation function on $D \subseteq \{0,1\}^n$ with $R(g) = \widetilde{\Omega}(\sqrt{n})$ and $Q(g) = O(1)$. Let $h : \{0,1\}^{m^2} \to \{0,1\}$ be the $\textsc{And-Or}_{m^2}$ function, defined as $\textsc{And}_m \circ \textsc{Or}_m$. This function satisfies $R(h) = \Omega(m^2)$, which can be shown using a variety of methods: it follows from the partition bound [JK10, Theorem 4] and also from our more general Theorem 5. We also have $C(h) = m$, since one 1 from each Or gate is a valid 1-certificate and an Or gate with all zero inputs is a valid 0-certificate. Lastly, $Q(h) = \widetilde{O}(m)$ follows from simply composing quantum algorithms for And and Or, and indeed $Q(h) = O(m)$ [HMdW03].

Let $f = g \circ h$ be a partial function on $nm^2$ bits. We have $Q(f) = O(m)$ by composition, since quantum algorithms can be composed in general without losing a log factor due to error reduction [Rei11, LMR+11]. There also exists a certificate of size $\widetilde{O}(nm)$ that proves that the input satisfies the promise, since we can certify the outputs of the $n$ different $h$ gates using $C(h) = m$ pointers to the relevant input bits. Since each pointer uses $O(\log n)$ bits, the certificate is of size $\widetilde{O}(nm)$. Also note that a quantum algorithm with *query access* to this certificate can check its validity using Grover search in $\widetilde{O}(\sqrt{nm})$ queries.

Lastly, we claim that $R(f) = R(g \circ h) = \Omega(R(g)R(h)) = \widetilde{\Omega}(\sqrt{n}m^2)$. While a general composition theorem for bounded-error or zero-error randomized query complexity is unknown, we can show such a result when the inner function is the Or function.

**Theorem 5.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq \{0,1\}^n$. Then $R(f \circ \mathrm{OR}_m) = \Omega(mR(f))$ and $R_0(f \circ \mathrm{OR}_m) = \Omega(mR_0(f))$. Therefore $R(f \circ \mathrm{AND}_m \circ \mathrm{OR}_m) = \Omega(m^2R(f))$ and $R_0(f \circ \mathrm{AND}_m \circ \mathrm{OR}_m) = \Omega(m^2R_0(f))$.*

*Proof.* We prove this for bounded-error algorithms; the argument for zero-error algorithms will be almost identical. Let $A$ be the best randomized algorithm for $f \circ \mathrm{OR}_m$. We convert $A$ into an algorithm $B$ for $f$, as follows.

Given input $x$ to $f$, $B$ generates $n$ random inputs to $\mathrm{OR}_m$, denoted by $Y_1, Y_2, \ldots, Y_n$, in the following way. First, each $Y_i$ is set to $0^m$, the all-zero string. Next, a random entry of $Y_i$ is chosen and replaced with a $*$ symbol. Finally, for each index $i$, the $*$ in $Y_i$ will be replaced by the bit $x_i$. This causes $\mathrm{OR}_m(Y_i) = x_i$ to be true for all $i$.

To evaluate $f(x)$, the algorithm $B$ can simply run $A$ on the string $Y_1Y_2 \ldots Y_n$. Since

$$f \circ \mathrm{OR}_m(Y_1Y_2 \ldots Y_n) = f(\mathrm{OR}_m(Y_1)\mathrm{OR}_m(Y_2) \ldots \mathrm{OR}_m(Y_n)) = f(x_1x_2 \ldots x_n) = f(x), \tag{2}$$

this algorithm evaluates $f(x)$ with the same error as $A$. This process uses $R(f \circ \mathrm{OR}_m)$ queries to the input $Y$. However, not all of these queries require $B$ to query $x$. In fact, $B$ only needs to query $x$ when algorithm $A$ queries a bit that was formerly a $*$ in one of the $Y_i$. The expected number of queries $B$ makes is therefore the expected number of $*$ entries found by the algorithm $A$. Using Markov's inequality, we can turn $B$ into an $R(f)$ algorithm, which uses a fixed number of queries to calculate $f(x)$ with bounded error; it follows that the expected number of $*$ entries found by $A$ is at least $\Omega(R(f))$.

We can now view $A$ as a randomized algorithm that finds $\Omega(R(f))$ star bits (in expectation) given input in $Y_1Y_2 \ldots Y_n$. Set $Y = Y_i$ for a randomly chosen $i$. The number of queries $A$ makes to $Y$ must be exactly $R(f \circ \mathrm{OR}_m)/n$ in expectation. Let the probability that $A$ finds $k$ stars be $p_k$, for $k = 0, 1, \ldots, n$. Then

$$\sum_{k=0}^{n} kp_k = \Omega(R(f)). \tag{3}$$

For each $k$, the probability that $A$ finds a star in $Y$ given it found $k$ stars in total is $k/n$. The overall probability that $A$ finds a star in $Y$ is therefore

$$\sum_{k=0}^{n} p_k(k/n) = \frac{1}{n}\sum_{k=0}^{n} kp_k = \Omega(R(f)/n). \tag{4}$$

In other words, $A$ makes $R(f \circ \mathrm{OR}_m)/n$ expected queries to $Y$, and finds the $*$ in $Y$ with probability $\Omega(R(f)/n)$. Now, finding the single $*$ in an otherwise all-zero string is an unordered search problem; the chance of solving this problem after $T$ queries is at most $T/m$. In other words, any decision tree of height $T$ has probability only $T/m$ of solving the problem. Since $A$'s querying strategy on $Y$ can be written as a probability distribution over decision trees with expected height $R(f \circ \mathrm{OR}_m)/n$, it follows that the probability of $A$ finding the $*$ is at most $R(f \circ \mathrm{OR}_m)/nm$. Thus we have $R(f \circ \mathrm{OR}_m)/nm = \Omega(R(f)/n)$, or $R(f \circ \mathrm{OR}_m) = \Omega(mR(f))$.

If $A$ is a zero-error randomized algorithm instead of a bounded-error algorithm, the same argument follows verbatim, except that there's no longer a need to use Markov's inequality to conclude that the expected number of stars found by $A$ is at least $\Omega(R_0(f))$.

The last part follows since we can show the same result for $\mathrm{AND}_m$, using the fact that the query complexity of a function $f(x_1, \ldots, x_n)$ equals that of $f(\overline{x_1}, \ldots, \overline{x_n})$ and the query complexity of $f$ and $\bar{f}$ is the same. □

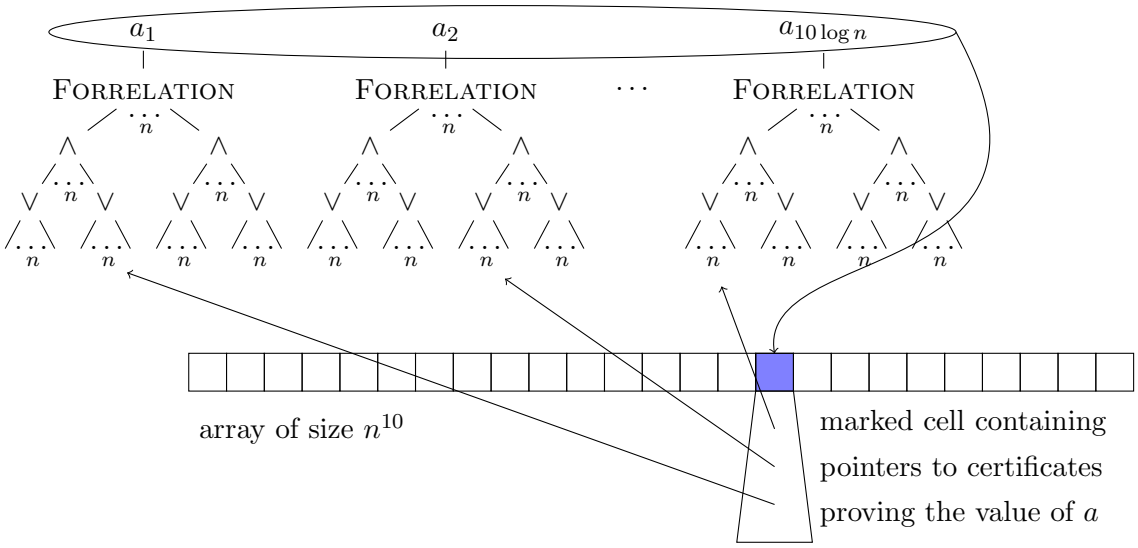


Figure 3: The function $f$ that achieves a superquadratic separation between $Q(f) = \widetilde{O}(n)$ and $R(f) = \widetilde{\Omega}(n^{2.5})$.

We now define the cheat sheet version of $f$, which we call $f_{\mathrm{CS}}$, depicted in Figure 3. Let the input to $f_{\mathrm{CS}}$ consist of $10 \log n$ inputs to $f$, each of size $nm^2$, followed by $n^{10}$ blocks of bits of size $\widetilde{O}(mn)$ each, which we refer to as the array of cells or array of cheat sheets. Each cell is large enough to hold certificates for all $10 \log n$ inputs to $f$ that certify for each input the output of $f$ evaluated on that input and that the promise holds.

Let us denote the input to $f_{\mathrm{CS}}$ as $z = (x^1, x^2, \ldots, x^{10 \log n}, Y_1, Y_2, \ldots, Y_{n^{10}})$, where $x^i$ is an input to $f$, and the $Y_i$ are the aforementioned cells of size $\widetilde{O}(mn)$. We define the value of $f_{\mathrm{CS}}(z)$ to be 1 if and only if the following conditions hold:

1. For all $i$, $x^i$ is in the domain of $f$. If this condition is satisfied, let $\ell$ be the positive integer corresponding to the binary string $(f(x^1), f(x^2), \ldots, f(x^{10 \log n}))$.
2. $Y_\ell$ certifies that all $x^i$ are in the domain of $f$ and that $\ell$ equals the binary string formed by their output values, $(f(x^1), f(x^2), \ldots, f(x^{10 \log n}))$.

We now upper bound the quantum query complexity of $f_{\mathrm{CS}}$. The quantum algorithm starts by assuming that the first condition of $f_{\mathrm{CS}}$ holds and simply computes $f$ on all $10 \log n$ inputs, which uses $\widetilde{O}(Q(f)) = \widetilde{O}(m)$ queries. The answers to these inputs points to the cheat sheet $Y_\ell$, where $\ell$ is the integer corresponding to the binary string $(f(x^1), f(x^2), \ldots, f(x^{10 \log n}))$. As discussed, verifying the cheat sheet of size $\widetilde{O}(mn)$ requires only $\widetilde{O}(\sqrt{mn})$ queries by a recursive application of Grover search. The algorithm outputs 1 if and only if the verification of $Y_\ell$ succeeded. If the certificate is accepted by the algorithm, then both conditions of $f_{\mathrm{CS}}$ are satisfied and hence it is easy to see the algorithm is correct. Hence $Q(f_{\mathrm{CS}}) = \widetilde{O}(m + \sqrt{mn})$.

We also know that $R(f) = \widetilde{\Omega}(\sqrt{n}m^2)$. We now prove that this implies $R(f_{\mathrm{CS}}) = \widetilde{\Omega}(R(f)) = \widetilde{\Omega}(\sqrt{n}m^2)$. Proving this completes the proof of Theorem 1, since setting $m = n$ immediately yields $Q(f_{\mathrm{CS}}) = \widetilde{O}(n)$ and $R(f_{\mathrm{CS}}) = \widetilde{\Omega}(n^{2.5})$.

To complete the proof, we show in general that $R(f_{\mathrm{CS}}) = \widetilde{\Omega}(R(f))$.

**Lemma 6.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$, and let $f_{\mathrm{CS}}$ be the cheat sheet version of $f$ with $c = 10 \log n$ copies of $f$. Then $R(f_{\mathrm{CS}}) = \Omega(R(f)/c^2) = \widetilde{\Omega}(R(f))$.*

*Proof.* Let $A$ be any bounded-error randomized algorithm for evaluating $f_{\mathrm{CS}}$. We will prove that

$A$ makes $\Omega(R(f)/c^2)$ queries.

We start by proving the following claim: Let $x^1, x^2, \ldots, x^c \in \mathrm{Dom}(f)$ denote $c$ valid inputs to $f$ and let $z$ be an input to $f_{\mathrm{CS}}$ consisting of $x^1x^2\cdots x^c$ with blank array—that is, it has 0 in all the entries of all cheat sheets. Let us assume that the all zero string is not a valid cheat sheet for any input. This can be enforced, for example, by requiring that all cheat sheets begin with the first bit equal to 1. Let $\ell$ be the binary number $f(x^1)f(x^2)\cdots f(x^c)$, and let $h_\ell$ be the $\ell^{\text{th}}$ cheat sheet. Then we claim that $A$ must query a bit of $h_\ell$ when run on $z$ with probability at least $1/3$.

This claim is easy to prove. Since the all zero string is an invalid cheat sheet, it follows that $f_{\mathrm{CS}}(z) = 0$, so $A$ outputs 0 when run on $z$ with probability at least $2/3$. Now if we modify $h_\ell$ in $z$ to be the correct cheat sheet for the given input $x^1, x^2, \ldots, x^c$, then we obtain a new input $z'$ with $f_{\mathrm{CS}}(z') = 1$. This means $A$ outputs 1 when run on $z'$ with probability at least $2/3$. However, since these inputs only differ on $h_\ell$, if $A$ does not query $h_\ell$ with high probability, it cannot distinguish input $z$ from input $z'$. Since $A$ is a valid algorithm for $f_{\mathrm{CS}}$, the claim follows.

We now use the hybrid argument to prove the lemma. For any input $z$ with blank array, let $p_z \in [0,1]^{2^c}$ be the vector such that for each $i = [2^c]$, $(p_z)_i$ is the probability that $A$ makes a query to the $i^{\text{th}}$ cheat sheet when given input $z$. Then the 1-norm of $p_z$ is at most the running time of $A$, which is at most $R(f)$. By the claim, if $\ell$ is the relevant cheat sheet for $z$, we have $(p_z)_\ell \geq 1/3$. On the other hand, since $p_z$ has $2^c \geq n^{10} \geq R(f)^{10}$ entries that sum to at most $R(f)$, almost all of them have value less than, say, $R(f)^{-5}$.

Next, consider hard distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ over the 0- and 1-inputs to $f$, respectively. We pick these distributions such that distinguishing between them with probability at least $1/2+1/12c$ takes at least $\Omega(R(f)/c^2)$ queries for any randomized algorithm.

For each $i \in [2^c]$, let $q_i$ be the expectation over $p_z$ when $z$ is made of $c$ inputs to $f$ generated from $\mathcal{D}^i = \mathcal{D}^{i_1} \times \mathcal{D}^{i_2} \times \cdots \times \mathcal{D}^{i_c}$, together with a blank array (here $i_j$ means the $j^{\text{th}}$ bit of $i$ when written in binary). Then for all $i \in [2^c]$, we have $(q_i)_i \geq 1/3$, and the 1-norm of $q_i$ is at most $R(f)$, so most entries of $q_i$ are less than $R(f)^{-5}$. The entries of $q_i$ can be interpreted as the probabilities of each cheat sheet being queried by the algorithm on an input sampled from $\mathcal{D}^i$.

Let $k \in [2^c]$ be such that $(q_0)_k < 1/6$. Let $k_0, k_1, \ldots, k_c$ be such that $k_0 = 0$, $k_c = k$, and consecutive $k$'s differ by at most one bit (when represented as binary vectors of length $c$). Since $(q_{k_0})_k < 1/6$ and $(q_{k_c})_k \geq 1/3$, there must be some $j \in [c]$ such that $(q_{k_{j+1}})_k - (q_{k_j})_k > 1/6c$ and $(q_{k_j})_k < 1/3$. Let $a = (q_{k_{j+1}})_k$ and let $b = (q_{k_j})_k$.

We can now use this to distinguish the product distribution $\mathcal{D}^{k_{j+1}}$ from $\mathcal{D}^{k_j}$, with probability of error at most $1/2 - 1/12c$: simply run the algorithm $A$, output 1 if it queried an entry of the $k^{\text{th}}$ cheat sheet, and otherwise output 1 with probability $\max(0, \frac{1-a-b}{2-a-b})$. If this maximum is 0, it means we have $b > 2/3$ and $a < 1/3$, so this algorithm has error at most $1/3$. Otherwise, it is not hard to check that the algorithm will determine if the input is from $\mathcal{D}^{k_{j+1}}$ with probability at most $1/2 - (b-a)/2 < 1/2 - 1/12c$.

Finally, note that since $k_{j+1}$ and $k_j$ differ in only one bit, distinguishing $\mathcal{D}^{k_{j+1}}$ from $\mathcal{D}^{k_j}$ allows us to distinguish between $\mathcal{D}^0$ and $\mathcal{D}^1$. This is because given any input from $\mathcal{D}^0$ or $\mathcal{D}^1$, the algorithm can sample other inputs to construct an input from $\mathcal{D}^{k_{j+1}}$ or $\mathcal{D}^{k_j}$, and then run the distinguishing algorithm. It follows that the running time of $A$ is at least $\Omega(R(f)/c^2)$, by the choice of the distributions $\mathcal{D}^0$ and $\mathcal{D}^1$. □

# 4 Quantum query complexity versus certificate complexity and degree

We now show a nearly quadratic separation between $Q(f)$ and $C(f)$, which yields a similar separation between $Q(f)$ and $\deg(f)$. We will also use the functions introduced in this section as building blocks to obtain the nearly $4^{\text{th}}$ power separation between $Q(f)$ and $\widetilde{\deg}(f)$ proved in Section 5.

## 4.1 Quadratic gap with certificate complexity

In this section we establish Theorem 3, restated for convenience:

**Theorem 3.** *There exists a total function $f$ such that $Q(f) = \widetilde{\Omega}(C(f)^2)$.*

Consider the $k$-SUM problem, $k\text{-SUM} : [M]^n \to \{0,1\}$, which asks if there are $k$ elements in the input string $x_1, x_2, \ldots, x_n \in [M]$ that sum to 0 (mod $M$). Belovs and Špalek [BŠ13] showed that $Q(k\text{-SUM}) = \Omega(n^{k/(k+1)})$ when the alphabet $M$ has size $n^k$ and $k$ is constant. In Appendix A, we show that their proof implies a bound of $Q(k\text{-SUM}) = \Omega(n^{k/(k+1)}/\sqrt{k})$ for super-constant $k$.

Now the 1-certificate complexity of the $k$-SUM problem is $k$ (assuming it costs one query to get an element of $M$), since it suffices to provide the $k$ elements that sum to 0 (mod $M$). If we take $k = \log n$, we get $Q(k\text{-SUM}) = \Omega(n/\sqrt{\log n})$ and $C_1(k\text{-SUM}) = O(\log n)$. Although this function is not Boolean, turning it into a Boolean function will only incur an additional polylogarithmic loss.

While $k$-SUM does not separate quantum query complexity from certificate complexity, its 1-certificate complexity is much smaller than its quantum query complexity. Composing a function with small 0-certificates, such as the $\text{AND}_n$ with $k$-SUM already gives a function whose quantum query complexity is larger than its certificate complexity: in this case, we have certificate complexity $\widetilde{O}(n)$ and quantum query complexity $\widetilde{\Omega}(n^{3/2})$, which follows from the following general composition theorem [HLŠ07, Rei11, LMR+11, Kim12]:

**Theorem 7** (Composition theorem for quantum query complexity)**.** *Let $f : D \to \{0,1\}$ and $g : E \to \{0,1\}$ be partial functions where $D \subseteq \{0,1\}^n$ and $E \subseteq \{0,1\}^m$. Then $Q(f \circ g) = \Theta(Q(f)Q(g))$.*

To get an almost quadratic gap between $Q(f)$ and $C(f)$, we use a variant of $k$-SUM itself as the outer function instead of the AND function. From $k$-SUM, we define a new Boolean function that we call BLOCK $k$-SUM, whose quantum query complexity is $\widetilde{\Theta}(n)$ and certificate complexity is also $\widetilde{\Theta}(n)$. However, although its certificate complexity is linear, the certificates consist almost exclusively of input bits set to 1 and only $\widetilde{O}(1)$ input bits set to 0. This means if we compose this function with $k$-SUM, the composed function has certificates of size $\widetilde{O}(n)$, since the certificates of BLOCK $k$-SUM are essentially composed of 1s, which are easy to certify for $k$-SUM. We denote this composed function $\text{BKK} : \{0,1\}^{n^2} \to \{0,1\}$. It satisfies $C(\text{BKK}) = \widetilde{O}(n)$ and $Q(\text{BKK}) = \widetilde{\Omega}(n^2)$, which yields the desired quadratic separation.

We now define the BLOCK $k$-SUM problem.

**Definition 8.** *Let* BLOCK $k$-SUM *be a total Boolean function on $n$ bits defined as follows. We split the input into blocks of size $10k \log n$ each and say a block is* balanced *if it has an equal number of 0s and 1s. Let the balanced blocks represent numbers in an alphabet $M$ of size $\Omega(n^k)$. The value of the function is 1 if and only if there are $k$ balanced blocks whose corresponding numbers sum to 0 (mod $M$) and all other blocks have at least as many 1s as 0s.*

We then compose this function with $k$-SUM to get the function BKK.

**Definition 9.** *Let* $\mathrm{BKK}_{n^2,k} : \{0,1\}^{n^2} \to \{0,1\}$ *be the function* Block $k$-sum *on* $n$ *bits composed with a Boolean version of* $k$-sum *on* $n$ *bits, and define* $\mathrm{BKK}_{n^2}$ *to be* $\mathrm{BKK}_{n^2,k}$ *with* $k = \log n$.

We are now ready to establish the various complexities of BKK.

**Theorem 10.** *For the total Boolean function* $\mathrm{BKK}_{n^2,k} : \{0,1\}^{n^2} \to \{0,1\}$, *we have*

$$C(\mathrm{BKK}_{n^2,k}) = O(k^2 n \log n) \quad \textit{and} \quad Q(\mathrm{BKK}_{n^2,k}) = \Omega\Big(\frac{n^{2-2/(k+1)}}{k^3 \log^2 n}\Big). \tag{5}$$

*Proof.* We start by analyzing the certificates of Block $k$-sum. The key property we need is that every input of Block $k$-sum has a certificate that uses very few 0s, but can use a large number of 1s. To see this, note that we can certify a 1-input by showing the $k$ balanced blocks that sum to 0 which requires $O(k^2 \log n)$ 0s, and all the 1s in every other block. There are two kinds of 0 inputs to certify: A 0-input that has a block with more 0s than 1s can be certified by providing that block, which only uses $O(k \log n)$ 0s. A 0-input in which all blocks have at least as many 1s as 0s can be certified by providing all the 1s: this provides the number represented by each block if it were balanced (though it does not prove the block is actually balanced), which is enough to check that no $k$ of them sum to zero. In conclusion, Block $k$-sum can always be certified by providing $O(n)$ 1s and $O(k^2 \log n)$ 0s.

We now analyze the certificate complexity of $\mathrm{BKK}_{n^2,k}$. For each input, the outer Block $k$-sum has a certificate using $O(k^2 \log n)$ 0s and $O(n)$ 1s. The inner function, $k$-sum, has 1-certificates of size $O(k^2 \log n)$ since there are $k$ numbers to exhibit and each uses $k \log n$ bits when represented in binary, and has 0-certificates of size $O(n)$. Therefore, the composed function always has a certificate of size $O(k^2 n \log n)$. Hence $C(\mathrm{BKK}_{n^2,k}) = O(k^2 n \log n)$.

The quantum query complexity of Block $k$-sum is $\Omega(Q(k\text{-sum})/k \log n)$ by a reduction from $k$-sum. Using the result in Appendix A, this is $\Omega(n^{1-1/(k+1)}/k^{3/2} \log n)$. Invoking the composition theorem for quantum query complexity (Theorem 7), we get $Q(\mathrm{BKK}_{n^2,k}) = \Omega\left(\frac{n^{2-2/(k+1)}}{k^3 \log^2 n}\right)$. ☐

Thus for the function $\mathrm{BKK} : \{0,1\}^{n^2} \to \{0,1\}$, defined as $\mathrm{BKK} = \mathrm{BKK}_{n^2,\log n}$, we have

$$C(\mathrm{BKK}) = O(n \log^3 n) = \widetilde{O}(n) \quad \text{and} \quad Q(\mathrm{BKK}) = \Omega\Big(\frac{n^2}{\log^5 n}\Big) = \widetilde{\Omega}(n^2). \tag{6}$$

This establishes Theorem 3, since BKK is a total Boolean function.

## 4.2 Quadratic gap with (exact) degree

We now show how to obtain a total function that nearly quadratically separates $Q(f)$ from $\deg(f)$ using any total function that achieves a similar separation between $Q(f)$ and $C(f)$. This proves Theorem 4:

**Theorem 4.** *There exists a total function* $f$ *such that* $Q(f) = \widetilde{\Omega}(\deg(f)^2)$.

Let $f : \{0,1\}^n \to \{0,1\}$ be a total function with $Q(f) = \widetilde{\Omega}(n)$ and $C(f) = \widetilde{O}(\sqrt{n})$, such as the BKK function introduced in the previous section. Let $f_{\mathrm{CS}}$ denote the cheat sheet version of $f$ (as described in Section 3), created using $10 \log n$ copies of $f$ that point to a cheat sheet among $n^{10}$ potential cheat sheets, where a valid cheat sheet contains certificates of size $\widetilde{O}(\sqrt{n})$ for each of the $10 \log n$ inputs to $f$ and the binary string corresponding to their outputs equals the location of the cheat sheet. In this case $f$ is a total function so the cheat sheets do not certify that the input

satisfies the promise, but only the value of $f$ evaluated on the input. We claim that the cheat sheet version of $f$ satisfies $Q(f_{\mathrm{CS}}) = \widetilde{\Omega}(n)$ and $\deg(f_{\mathrm{CS}}) = \widetilde{O}(\sqrt{n})$.

Let us start with the degree upper bound, $\deg(f_{\mathrm{CS}}) = \widetilde{O}(\sqrt{n})$. Let $\ell \in [n^{10}]$ be a potential location of the cheat sheet. For any $\ell$, consider the problem of outputting 1 if and only if $f_{\mathrm{CS}}(z) = 1$ and $\ell$ is the location of the cheat sheet for the input $z$. Since $\ell$ is known, this can be solved by a deterministic algorithm $\mathcal{A}_\ell$ that makes $\widetilde{O}(\sqrt{n})$ queries, since it can simply check if the certificate stored at cell $\ell$ in the array is valid: it can check all the $10 \log n$ certificates of size $\widetilde{O}(\sqrt{n})$ for each of the $10 \log n$ instances of $f$ and then check if the outputs of $f$ evaluate to the location $\ell$. Since polynomial degree is at most deterministic query complexity, we can construct a representing polynomial for $\mathcal{A}_\ell$ for any location $\ell$. This is a polynomial $p_\ell$ of degree $\widetilde{O}(\sqrt{n})$ such that $p_\ell(z) = 1$ if and only if $f_{\mathrm{CS}}(z) = 1$ and $\ell$ is the position of the cheat sheet on input $z$. Now we can simply add all the polynomials $p_\ell$ together to obtain a new polynomial $q$ of the same degree. We claim $q(z) = f_{\mathrm{CS}}(z)$ since if $f_{\mathrm{CS}}(z) = 0$ then certainly all the polynomials $p_\ell(z) = 0$ (since none of the cheat sheets is valid) and if $f_{\mathrm{CS}}(z) = 1$ then $q(z) = 1$ because exactly one of many $p_\ell(z)$ will evaluate to 1, the one corresponding to the location of the cheat sheet for the input $z$. Note that the property used here is that in a 1-input to $f_{\mathrm{CS}}$, exactly one location serves as the correct cheat sheet, i.e., the location of the cheat sheet for a 1-input is unique.

The claim that $Q(f_{\mathrm{CS}}) = \widetilde{\Omega}(n)$ seems intuitive since $Q(f) = \widetilde{\Omega}(n)$ and the cheat sheet version of $f$ cannot be easier than $f$ itself. This intuitive claim is true and we show below that $Q(f_{\mathrm{CS}}) = \Omega(Q(f))$ in general, which completes the proof of Theorem 4.

To prove this general result for quantum query complexity, we will need the following strong direct product theorem due to Lee and Roland [LR13].

**Theorem 11.** *Let $f$ be a (partial) function with $Q_{1/4}(f) \geq T$. Then any $T$-query quantum algorithm that outputs the value of $f$ evaluated on $c$ independent instances has success probability at most $O((3/4)^{c/2})$.*

We now prove the lower bound on the quantum query complexity of cheat sheet functions.

**Lemma 12.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$, and let $f_{\mathrm{CS}}$ be a cheat sheet version of $f$ with $c = 10 \log n$ copies of $f$. Then $Q(f_{\mathrm{CS}}) = \Omega(Q(f))$.*

*Proof.* By Theorem 11, we know that given $c = 10 \log n$ instances of $f$, any quantum algorithm that makes fewer than $T = Q_{1/4}(f)$ queries will correctly decide all the instances with probability at most $O((3/4)^{c/2})$.

Now suppose by way of contradiction that $Q_{1/4}(f_{\mathrm{CS}}) \leq T$. Then we will show how to decide $c$ copies of $f$ with probability $\Omega\left(1/T^2\right)$ by making $T$ quantum queries. Since

$$\frac{1}{T^2} \geq \frac{1}{n^2} \geq \frac{1}{2^{c/5}} \gg \left(\frac{3}{4}\right)^{c/2}, \tag{7}$$

this is enough to contradict Theorem 11.

Let $Q$ be a quantum algorithm that decides $f_{\mathrm{CS}}$ using at most $T$ queries. Then consider running $Q$ on an input $z = (x^1, \ldots, x^c, Y_1, Y_2, \ldots, Y_{2^c})$, where the $x^i$ are in the domain of $f$ and whose cheat-sheet array $(Y_1, Y_2, \ldots, Y_{2^c})$ has been completely zeroed out. We assume again that the all-zero cheat sheet is invalid for any input. This can always be enforced by requiring a valid cheat sheet to have the first bit set to 1.

For all $y \in [2^c]$ and $t \in [T]$, define $m_{y,t}$, or *the query magnitude on $y$ at query $t$*, to be the probability that $Q$ would be found querying some bit in cell $Y_y$, were we to measure in the standard basis immediately before the $t^{\text{th}}$ query.

For an input $z$ where we have zeroed out the cheat sheet array, clearly $f_{\mathrm{CS}}(z) = 0$ and hence $Q$ outputs 0 on this input with high probability. On the other hand, if we let $\ell = f\left(x^1\right) \cdots f\left(x^c\right) \in [2^c]$, by modifying only the $\ell^{\text{th}}$ cell in the array, $Y_\ell$, we could produce an input $z'$ such that $f_{\mathrm{CS}}\left(z'\right) = 1$. From these facts, together with the standard BBBV hybrid argument [BBBV97], it follows that

$$\sqrt{m_{\ell,1}} + \cdots + \sqrt{m_{\ell,T}} = \Omega\left(1\right). \tag{8}$$

So by the Cauchy–Schwarz inequality, we have

$$m_{\ell,1} + \cdots + m_{\ell,T} = \Omega\left(\frac{1}{T}\right). \tag{9}$$

This implies that, if we simply choose a $t \in [T]$ uniformly at random, run $Q$ until the $t^{\text{th}}$ query with the cheat-sheet array zeroed out, and then measure in the standard basis, we will observe $\ell = f\left(x^1\right) \cdots f\left(x^c\right)$ with probability $\Omega\left(1/T^2\right)$. This completes the proof. □

# 5 Quantum query complexity versus approximate degree

We now show how to obtain a nearly quartic separation between quantum query complexity and approximate degree from a function that quadratically separates $Q(f)$ from $C(f)$. As in Section 3, we first motivate the construction in Section 5.1 and then formally give the separation in Section 5.2. The main result of this section is Theorem 2:

**Theorem 2.** *There exists a total function $f$ such that $Q(f) \geq \widetilde{\deg}(f)^{4-o(1)}$.*

## 5.1 Intuition

To obtain the quartic separation with approximate degree, we could try the same approach as in the previous section for exact degree. Using notation from Section 4.2, consider the algorithm $\mathcal{A}_\ell$ that makes $\widetilde{O}(\sqrt{n})$ queries. Instead of using a polynomial to exactly represent $\mathcal{A}_\ell$, we could try to construct more efficient approximating polynomials. If there were a quantum algorithm that checked the certificate quadratically faster than the deterministic $\mathcal{A}_\ell$, then we would be done. This is because such a quantum algorithm would yield a polynomial of degree $\widetilde{O}(n^{1/4})$ that approximates $\mathcal{A}_\ell$. Then we could sum up these polynomials as before, with some error reduction to ensure that the error in each polynomial is small enough not to affect the sum of the polynomials. This error reduction only adds an overhead of $O(\log n)$, which gives us a polynomial of degree $\widetilde{O}(n^{1/4})$ that approximates $f_{\mathrm{CS}}$.

However, it is unclear if there is a quantum algorithm to check a certificate of size $\widetilde{O}(\sqrt{n})$ quickly. Reading the certificate itself could require $\widetilde{\Omega}(\sqrt{n})$ queries. All we know is that the certificate can be checked using $\widetilde{O}(\sqrt{n})$ queries classically, but this does not imply a quadratic quantum speedup. To obtain a quadratic speedup the certificates need some structure that can, for example, be exploited with Grover's algorithm. But the certificates in this case are simply 0-inputs of $k$-SUM of size $\widetilde{O}(\sqrt{n})$, and it is unclear how to quantumly check if an input is a 0-input (even with query access to any certificate) any quicker than querying the entire input. What we would like is for the certificate to be checkable using $\widetilde{O}(n^{1/4})$ queries to the certificate and the input. So we construct a new function that has this property by modifying the BKK function used in Section 4.2.

Let $f$ be BKK : $\{0,1\}^{n^2} \to \{0,1\}$ for some large, but constant $n$. We choose $n$ to be large enough that $\mathrm{Adv}^{\pm}(f) \geq n^{1.99}$ and $C(f) \leq n^{1.01}$. Such an $n$ exists because asymptotically

$C(\mathrm{BKK}) = \widetilde{O}(n)$ and $Q(\mathrm{BKK}) = \widetilde{\Omega}(n^2)$ (and $Q(\mathrm{BKK}) = \Theta(\mathrm{Adv}^{\pm}(\mathrm{BKK}))$ [LMR+11]). Composing this function with itself $d$ times gives us a new function $g : \{0,1\}^N \to \{0,1\}$, where $N = n^{2d}$. This function has $C(g) \leq C(f)^d \leq n^{1.01d} \leq N^{0.51}$. Since the general adversary bound satisfies a perfect composition theorem [LMR+11], we have $Q(g) = \Omega(\mathrm{Adv}^{\pm}(g)) = \Omega(\mathrm{Adv}^{\pm}(f)^d) = \Omega(n^{1.99d}) = \Omega(N^{0.99})$.

If we view the function $g$ as a tree of depth $d$ and fanin $n^2$, it has $N$ leaves but only $N^{0.51}$ of these leaves are part of a certificate. Consider the subtree of $g$ that consists of all nodes that are ancestors of these $N^{0.51}$ certificate leaves. This subtree has size $O(N^{0.51})$ and the set of values of all nodes in this subtree is a certificate for $g$. Furthermore, this certificate can be checked by a quantum algorithm in only $O(N^{0.26})$ queries, since each level of the tree consists of at most $N^{0.51}$ instances of $f$ acting on a constant number of bits. Checking a constant-sized $f$ costs only $O(1)$ queries, and searching over all $N^{0.51}$ instances for an invalid certificate costs $O(N^{0.26})$ queries by Grover's algorithm. This can be done for each level, resulting in a quantum algorithm with query complexity $\widetilde{O}(N^{0.26})$.

Thus we have constructed a function $g$ for which $Q(g)$ is close to $N$, but the complexity of quantumly checking a certificate (given query access to it) is close to $N^{1/4}$. Plugging this into the cheat sheet construction yields a nearly $4^{\text{th}}$ power separation between quantum query complexity and approximate degree.

### 5.2 Implementation

Recall the function $\mathrm{BKK} : \{0,1\}^{n^2} \to \{0,1\}$ introduced in Theorem 10. We introduce a function we call RecBKK which consists of recursively composing BKK with itself $d$ times; that is, we replace each bit in BKK with a new copy of BKK, then replace each bit in the new copies with yet more copies of BKK, and so on. The resulting function will look like a tree of height $d$ where each vertex has $n^2$ children and will have total input size $N = n^{2d}$. We will choose $d = (4/25)\log n / \log\log n$ to optimize our construction. The resulting function RecBKK has the following properties.

**Theorem 13.** *There exists a total function* $\mathrm{RecBKK} : \{0,1\}^N \to \{0,1\}$ *such that given query access to a certificate of size* $N^{1/2+o(1)}$*, a quantum algorithm can check the validity of the certificate using at most* $N^{1/4+o(1)}$ *queries. Furthermore,* $Q(\mathrm{RecBKK}) = N^{1-o(1)}$.

*Proof.* Our function RecBKK is defined as the $d$-fold composition of $\mathrm{BKK} : \{0,1\}^{n^2} \to \{0,1\}$ with itself. This yields a function on $N = n^{2d}$ bits and we set $d = (4/25)\log n / \log\log n$.

With this choice of $d$, we can show more precisely that the function RecBKK has a certificate that can be checked by a quantum algorithm using $O(N^{1/4}L_N[1/2,1]) = N^{1/4+o(1)}$ queries, where $L_N[a,c] = \exp((c+o(1))\log^a N \log\log^{1-a} N)$, and $Q(\mathrm{RecBKK}) = \Omega(N/L_N[1/2,1]) = N^{1-o(1)}$.

With this choice of $d$, the parameters are now related as follows: Since $N = n^{2d}$, we have $\log N = 2d\log n = (8/25)\log^2 n / \log\log n$. This gives $\log n = (5/4+o(1))\sqrt{\log N \log\log N}$, so we get $n = L_N[1/2, 5/4]$. Additionally, since $(\log n)^d = \exp(d\log\log n) = \exp((4/25)\log n) = n^{4/25}$, it follows that $(\log n)^d = L_N[1/2, 1/5]$.

We start with the quantum lower bound for $Q(\mathrm{RecBKK})$. By Theorem 10 and the optimality of the general adversary bound [LMR+11], we have

$$\mathrm{Adv}^{\pm}(\mathrm{BKK}) = \Omega(Q(\mathrm{BKK})) = \Omega\left(\frac{n^2}{\log^5 n}\right) = \frac{n^2}{(\log n)^{5+o(1)}}. \tag{10}$$

By using the fact that $\mathrm{Adv}^{\pm}(f^d) = \mathrm{Adv}^{\pm}(f)^d$ for Boolean functions $f$ [LMR+11], we get

$$Q(\textsc{RecBKK}) = \Omega(\mathrm{Adv}^{\pm}(\mathrm{BKK})^d) = \frac{n^{2d}}{(\log n)^{(5+o(1))d}} = \frac{N}{L_N[1/2,1/5]^{5+o(1)}} = \Omega(N/L_N[1/2,1]). \tag{11}$$

Now we show the upper bound on quantumly checking a certificate. First note that every non-leaf node in the RecBKK tree corresponds to a BKK instance. For each such node, there is therefore a set of $C(\mathrm{BKK})$ children that constitute a certificate for that BKK instance. We can therefore certify the RecBKK instance by starting from the top of the tree, listing out $C(\mathrm{BKK})$ children that constitute a certificate, then listing out $C(\mathrm{BKK})$ children for each of those, and so on. In each layer $i$ of the tree, we thus have $C(\mathrm{BKK})^i$ nodes that belong to a certificate.

We require our quantumly checkable certificate to provide, for each non-leaf node that belongs to a certificate, pointers to $C(\mathrm{BKK})$ of the node's children that constitute a certificate for that BKK instance, starting with the root of the tree. A quantum algorithm can then use Grover search to search for a bad certificate. More precisely, the algorithm checks the certificate for the root to see if the $C(\mathrm{BKK})$ children of the root pointed to and their claimed values do indeed form a certificate. It then checks if all the claimed values in the first level are correct assuming the claimed values of nodes in the second level and so on. The total number of certificates to check is

$$1 + C(\mathrm{BKK}) + C(\mathrm{BKK})^2 + \cdots + C(\mathrm{BKK})^{d-1} \leq 2C(\mathrm{BKK})^{d-1}, \tag{12}$$

where each certificate has size $C(\mathrm{BKK})$. Therefore, the quantum algorithm will make

$$\widetilde{O}(C(\mathrm{BKK})^{(d-1)/2} C(\mathrm{BKK})) = \widetilde{O}(C(\mathrm{BKK})^{(d+1)/2}) \tag{13}$$

queries, where we have logarithmic factors due to the pointers being encoded in binary and error reduction. From Theorem 10, we have $C(\mathrm{BKK}) = O(n \log^3 n) = n(\log n)^{3+o(1)}$, so the search takes

$$\widetilde{O}(n^{(d+1)/2}(\log n)^{(3+o(1))(d+1)/2}) = \widetilde{O}(N^{1/4} n^{1/2} (\log n)^{(3/2+o(1))d}) = \widetilde{O}(N^{1/4} L_N[1/2, 37/40]). \tag{14}$$

quantum queries, which is is $O(N^{1/4} L_N[1/2,1])$. □

Using this function RecBKK we can now establish Theorem 2. This proof is very similar to the separation between quantum query complexity and exact degree in Section 4.2.

Let $f$ be the total function $\textsc{RecBKK} : \{0,1\}^n \to \{0,1\}$ with $Q(f) = n^{1-o(1)}$ and $C(f) = n^{1/2+o(1)}$, and more importantly given query access to this certificate, a quantum algorithm can check its validity using $n^{1/4+o(1)}$ queries. Let $f_{\mathrm{CS}}$ denote the cheat sheet version of $f$ created using $10 \log n$ copies of $f$ as before. From Lemma 12 we know that $Q(f_{\mathrm{CS}}) = \Omega(Q(f))$ and hence $Q(f_{\mathrm{CS}}) = n^{1-o(1)}$. We now show $\widetilde{\deg}(f_{\mathrm{CS}}) = n^{1/4+o(1)}$.

Let $\ell \in [n^{10}]$ be a potential location of the cheat sheet. For any $\ell$, consider the problem of outputting 1 if and only if $f_{\mathrm{CS}}(z) = 1$ and $\ell$ is the location of the cheat sheet for the input $z$. For any fixed $\ell$, this can be solved by a quantum algorithm $\mathcal{Q}_\ell$ that makes $n^{1/4+o(1)}$ queries as shown in Theorem 13, since it can simply check if the certificate stored at cell $\ell$ in the array is valid: it can check all the $10 \log n$ certificates for each of the $10 \log n$ instances of $f$ and then check if the outputs of $f$ evaluate to the location $\ell$.

Since approximate polynomial degree is at most quantum query complexity, we can construct a representing polynomial for $\mathcal{Q}_\ell$ for any location $\ell$. This is a polynomial $p_\ell$ of degree $n^{1/4+o(1)}$ such that $p_\ell(z) = 1$ if and only if $f_{\mathrm{CS}}(z) = 1$ and $\ell$ is the position of the cheat sheet on input $z$. Now we can simply add all the polynomials $p_\ell$ together to obtain a new polynomial $q$ of the same

degree, except that we first reduce the error in each polynomial to below $1/n^{10}$ so that the total error is bounded. (This can be done, for example, using the amplification polynomial construction of [BNRdW07, Lemma 1].) Now $q(z) = f_{\mathrm{CS}}(z)$ as argued in Section 4.2.

# 6 Cheat sheet functions

In this section we define the cheat sheet framework more generally. Once the framework is set up, proofs based on the framework are short and conveniently separate out facts about the framework from results about the separation under consideration. To demonstrate this, we reprove some known separations in Section 6.4.

## 6.1 Certifying functions and cheat sheets

Intuitively, a certifying function for a function $f : D \to \{0,1\}$, where $D \subseteq [M]^n$, is a function $\phi$ that takes in an input $x \in [M]^n$, a claimed value $y_1$ for $f(x)$, and a proof $(y_2, y_3, \ldots, y_k)$ that we indeed have $x \in D$ and $f(x) = y_1$. The value of the certifying function will be 1 only when these conditions hold. We would also like the certifying function to depend nontrivially on the certificate $y$, i.e., for every $x \in D$ there should be some $y$ that makes the function output 1, and some $y$ that makes it output 0. For convenience of analysis, we enforce that the certificate $y = 0^k$ is invalid for all $x$. Any nontrivial certifying function can be made to have this property by requiring that (say) the second bit of $y$, $y_2$, is 1 for all valid certificates.

**Definition 14** (Certifying function)**.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$ for some integer $M \geq 2$. We say a total function $\phi : [M]^n \times \{0,1\}^k \to \{0,1\}$, where $k$ is any positive integer, is a* certifying function *for $f$ if the following conditions are satisfied:*

1. $\forall x \notin D$, $\phi(x, y) = 0$ *(invalid inputs should not have certificates)*
2. $\forall x \in [M]^n$, $\forall y \in \{0,1\}^k$, *if* $y_1 \neq f(x)$ *then* $\phi(x, y) = 0$ *(certificate asserts $y_1 = f(x)$)*
3. $\forall x \in D$, $\exists y \in \{0,1\}^k$ *such that* $\phi(x, y) = 1$ *(valid inputs should have certificates)*
4. $\forall x \in [M]^n$, $\phi(x, 0^k) = 0$ *(nontriviality condition)*

Typically, a certifying function will be defined so that its value is only 1 if $y$ includes pointers to a certificate in $x$ and if $f$ is a partial function, we will also want $y$ to include a proof that $x$ satisfies the promise of $f$. Thus the query complexity of $\phi$ may be smaller than that of $f$.

We now define the cheat sheet version of a function $f$ with certifying function $\phi$. In the separations in the previous sections we used $10 \log n$ copies of the function $f$ to create the address of the cheat sheet. For generality we now use $c = \lceil 10 \log D(f) \rceil$ instead so that the construction only adds logarithmic factors in $D(f)$, as opposed to logarithmic factors in $n$, which may be larger than $D(f)$. This does not make a difference in our applications, however.

**Definition 15** (Cheat sheet version of $f$)**.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$ for some integer $M \geq 2$, and let $c = \lceil 10 \log D(f) \rceil$. Let $\phi : [M]^n \times \{0,1\}^k \to \{0,1\}$, for some positive integer $k$, be a certifying function for $f$. Then* the cheat sheet version of $f$ with respect to $\phi$, *denoted $f_{\mathrm{CS}(\phi)}$, is a total function*

$$f_{\mathrm{CS}(\phi)} : ([M]^n)^c \times ((\{0,1\}^k)^c)^{2^c} \to \{0,1\} \tag{15}$$

*acting on an input $(X, Y_1, Y_2, \ldots, Y_{2^c})$, where $X \in ([M]^n)^c$ and $Y_i \in (\{0,1\}^k)^c$. Also, let $X \in ([M]^n)^c$ be written as $X = (x^1, x^2, \ldots, x^c)$, where $x^i \in [M]^n$. Then we define the value of $f_{\mathrm{CS}(\phi)}(X, Y_1, Y_2, \ldots Y_{2^c})$ to be 1 if and only if conditions (1) and (2) hold.*

*(1) For all $i \in [c]$, $x^i \in D$.*

*If condition (1) is satisfied, let $\ell \in [2^c]$ be the positive integer corresponding to the binary string $(f(x^1), f(x^2), \ldots, f(x^c))$ and let $Y_\ell = (y^1, y^2, \ldots y^c)$ where each $y^i \in \{0,1\}^k$.*

*(2) For all $i \in [c]$, $x^i \in D$ and $\phi(x^i, y^i) = 1$.*

Intuitively, the input to $f_{\mathrm{CS}(\phi)}$ is interpreted as $c$ inputs to $f$, which we call $(x^1, x^2, \ldots, x^c)$, followed by $2^c$ strings $(Y_1, Y_2, \ldots Y_{2^c})$ of length $ck$ called cheat sheets. The value of the function will be 0 if any of the $c$ inputs to $f$ do not satisfy the promise of $f$. If they do satisfy the promise, let $\ell \in [2^c]$ be the number encoded by the binary string $f(x^1)f(x^2)\ldots f(x^c)$, where $0^c$ and $1^c$ encode the numbers 1 and $2^c$ respectively. If the $\ell^{\text{th}}$ cheat sheet in the input, $Y_\ell$, is written as $Y_\ell = (y^1, y^2, \ldots, y^c)$, where $y^i \in \{0,1\}^k$ for all $i$, then the value of $f_{\mathrm{CS}(\phi)}$ is 1 if $\phi(x^i, y^i) = 1$ for all $i \in [c]$, and 0 otherwise.

In other words, the value of $f_{\mathrm{CS}(\phi)}$ is 1 if the first part of its input consists of $c$ valid inputs to $f$ that together encode a location of a cheat sheet in the second part of the input, and this cheat sheet in turn contains pointers that certify the values of the $c$ inputs to $f$. The idea is that the only way to find the cheat sheet is to solve the $c$ copies of $f$, so that $f_{\mathrm{CS}(\phi)}$ has query complexity at least that of $f$, but once the cheat sheet has been found one can verify the value of $f_{\mathrm{CS}(\phi)}$ using only the query complexity of $\phi$, which may be much smaller.

We now characterize the query complexity of $f_{\mathrm{CS}(\phi)}$ in terms of the query complexities of $f$ and $\phi$ in various models. The following result summarizes our results for the query complexity of cheat sheet functions.

**Theorem 16** (Query complexity of cheat sheet functions)**.** *Let $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$, be a partial function, and let $\phi$ be a certifying function for $f$. Then the following relations hold.*

- $D(f_{\mathrm{CS}(\phi)}) = \widetilde{\Theta}(D(f) + D(\phi))$
- $R(f_{\mathrm{CS}(\phi)}) = \widetilde{\Theta}(R(f) + R(\phi))$
- $Q(f_{\mathrm{CS}(\phi)}) = \widetilde{\Theta}(Q(f) + Q(\phi))$
- $\deg(f_{\mathrm{CS}(\phi)}) = \widetilde{\Theta}(\deg(\phi))$
- $\widetilde{\deg}(f_{\mathrm{CS}(\phi)}) = \widetilde{\Theta}(\widetilde{\deg}(\phi))$
- $R_0(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(R_0(f) + R_0(\phi))$ *(but the corresponding upper bound relation is false)*
- $Q_E(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(Q_E(f) + Q_E(\phi))$ *(we conjecture that the corresponding lower bound holds)*

The algorithmic measures, $D$, $R$, and $Q$ behave as expected since Theorem 16 asserts that the straightforward way to compute $f_{\mathrm{CS}(\phi)}$ in these models, which is to first compute all $c$ copies of $f$ and then check if they point to a valid cheat sheet, is essentially optimal. We conjecture that $Q_E$, the quantum analogue of $D$, should behave similarly to $D$ or $Q$, but we can only prove the upper bound (see Lemma 17). In fact, if the lower bound for $Q_E$ can be proved, then we would get a near cubic separation between $Q(f)$ and $Q_E(f)$.

For the zero-error class $R_0$, while we can show that the lower bound holds, the upper bound relation is false (see Theorem 28 for a counterexample). We believe the zero-error class $Q_0$ behaves similarly: we conjecture the analogous lower bound holds for $Q_0$, but are unable to prove it. For $Q_0$, the corresponding upper bound is false, for reasons similar to those for $R_0$.

Notably, one-sided error measures do not behave well for cheat sheet functions at all. In Theorem 27, we demonstrate $f$ and $\phi$ such that $R_1(f_{\mathrm{CS}(\phi)}) \ll R_1(f)$, showing that the lower bound fails. The upper bound need not hold either.

Lastly, note that deg and $\widetilde{\deg}$ behave fundamentally differently on cheat sheet functions than the algorithmic measures. The (approximate) degree of the function $f$ does not even play a role in determining the (approximate) degree of $f_{\mathrm{CS}(\phi)}$.

Theorem 16 is proved in the next two sections. In Appendix B we present some examples where the naïvely expected relations do not hold.

### 6.2 Upper bounds on the complexity of cheat sheet functions

We start by showing that the usual algorithmic models can compute $f_{\mathrm{CS}(\phi)}$ in roughly the number of queries they require for $f$ and $\phi$.

**Lemma 17.** *Let $f : D \to \{0,1\}$, where $D \subseteq [M]^n$, be a partial function, and $\phi$ be a certifying function for $f$. Then the following upper bounds hold.*

- $D(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(D(f) + D(\phi))$
- $R(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(R(f) + R(\phi))$
- $Q(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(Q(f) + Q(\phi))$
- $Q_E(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(Q_E(f) + Q_E(\phi))$

*Proof.* The algorithm to evaluate $f_{\mathrm{CS}(\phi)}$ is the following: First, evaluate $f$ on the $c$ inputs to $f$ contained in $z$, assuming they satisfy the promise of $f$. If one of these inputs is discovered to contradict the promise of $f$, then output 0 and halt. This takes $cD(f)$ queries for a deterministic algorithm (and $cQ_E(f)$ queries for an exact quantum algorithm), $O(R(f)c\log c)$ for a bounded-error randomized algorithm, and $O(Q(f)c\log c)$ for a bounded-error quantum algorithm, where the factor of $O(\log c)$ comes from reducing the error enough to ensure that all $c$ functions are computed correctly with high probability. From the $c$ outputs we get a binary number $\ell$. We can then go to the appropriate cheat sheet, and evaluate $\phi$ on all the $(x^i, y^i)$ pairs and output 1 if they are all 1. This takes $cD(\phi)$ queries for a deterministic algorithm (and $cQ_E(\phi)$ queries for an exact quantum algorithm), $O(R(\phi)c\log c)$ queries for a randomized algorithm, and $O(Q(\phi)c\log c)$ queries for a quantum algorithm. Since $\phi(x, y) = 0$ when $x$ does not satisfy the promise of $f$, this algorithm is correct. □

Before proving the upper bounds for the degree measures, we prove a technical lemma that shows that $\log(D(f))$ and $\log(C(\phi))$ or $\log(\widetilde{\deg}(\phi))$ are the same up to constants. We prove this to show that the log factors that appear in our construction really can be neglected compared to the measures we consider.

**Lemma 18.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$, and let $\phi$ be a certifying function for $f$. Then $D(f) \leq C^{(1)}(\phi)^2$ and $C(f) \leq C^{(1)}(\phi)$. As a consequence, we have $D(f) \leq D(\phi)^2$ and $C(f) \leq C(\phi)$.*

*Proof.* Note that every 1-certificate for $\phi(x, y)$ proves that $f(x) = y_1$. In particular, such a 1-certificate must reveal a certificate for $f$ in $x$. This already shows that $C(f) \leq C^{(1)}(\phi)$.

Moreover, every 1-certificate of $\phi(x, y)$ where $y_1 = 0$ must conflict with every 1-certificate of $\phi(x, y)$ where $y_1 = 1$ on some bit in $x$. This is because otherwise, there would be some $x$ in the promise of $f$ that can be certified to be both a 0- and a 1-input for $f$.

We can then use the following strategy to compute $f(x)$. On input $x \in \mathrm{Dom}(f)$, pick a 1-certificate of some $\phi(x, y)$ with $y_1 = 0$, and query all of its entries in $x$. This reveals at least one bit of every 1-certificate of $\phi$ that has $y_1 = 1$. We then find another 1-certificate of some $\phi(x, y)$ where $y_1 = 0$, consistent with the inputs revealed in $x$ so far. We once again reveal all of its entries in $x$. After $C^{(1)}(\phi)$ iterations of this process, we have either eliminated all 1-certificates of $\phi$ with $y_1 = 0$, or else all 1-certificates of $\phi$ with $y_1 = 1$. This tells us the value of $f(x)$, because since $x \in \mathrm{Dom}(f)$, there must be some $y$ such that $\phi(x, y) = 1$, and hence there must be some 1-certificate of $\phi$ consistent with $x$. The total number of queries used was at most $C^{(1)}(\phi)^2$. □

Next, we show the upper bounds on the degree measures of $f_{\mathrm{CS}(\phi)}$ and show that they only depend on the degree of $\phi$ and not on any measure of the function $f$.

**Lemma 19.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq \{0,1\}^n$, and $\phi$ be a certifying function for $f$. Then $\deg(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(\deg(\phi))$ and $\widetilde{\deg}(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(\widetilde{\deg}(\phi))$.*

*Proof.* We start by showing this for $\deg(f_{\mathrm{CS}(\phi)})$. For each $\ell \in [2^c]$, we construct a polynomial which is 1 if $f_{\mathrm{CS}(\phi)}(z) = 1$ and the cheat sheet of $z$ is the $\ell^{\text{th}}$ cheat sheet, and 0 otherwise. We can do this by simply multiplying together the polynomials of $\phi(x^i, y^i)$ for all $i \in [c]$, where $x^i$ are the inputs to $f$ in $z$ and $y^i$ are the entries found in the $\ell^{\text{th}}$ cheat sheet of $z$.

The resulting polynomial, $p_\ell$, has degree $c\deg(\phi)$, is 1 on input $z$ if $z$ is a 1-input for $f_{\mathrm{CS}(\phi)}$ and $\ell$ is the location of the cheat sheet for input $z$, and is 0 on all other inputs. To get a polynomial for $f_{\mathrm{CS}(\phi)}$, we simply sum up the polynomials $p_\ell$ for all $\ell \in [2^c]$. The resulting polynomial is equal to $f_{\mathrm{CS}(\phi)}(z)$ on all inputs $z$, and has degree $c\deg(\phi)$.

The same trick works for $\widetilde{\deg}$, except in that case we cannot simply construct the AND of $c$ polynomials by multiplying them together: we must first perform error reduction on the polynomials and map the output range $[0,1/3] \cup [2/3,1]$ to $[0,1/3c] \cup [1-1/3c,1]$. It is possible to do this amplification while increasing the degree of the polynomials by at most $O(\log c)$ using, for example, the amplification polynomial construction of [BNRdW07, Lemma 1]. Therefore we have $\widetilde{\deg}(f_{\mathrm{CS}(\phi)}) = O(c\,\widetilde{\deg}(\phi)\log c) = \widetilde{O}(\widetilde{\deg}(\phi))$.

Finally, note that $c$ is at most logarithmic in $\widetilde{\deg}(\phi)$, since $c = O(\log D(f)) = O(\log C^{(1)}(\phi)) = O(\log C(\phi)) = O(\log \widetilde{\deg}(\phi))$, where the last equality follows from [BdW02] and from the fact that $\phi$ is a total function. □

### 6.3 Lower bounds on the complexity of cheat sheet functions

**Lemma 20.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq \{0,1\}^n$, and let $\phi$ be a certifying function for $f$. Then the complexity of $f_{\mathrm{CS}(\phi)}$ is at least that of $\phi$ with respect to any complexity measure that does not increase upon learning input bits, which includes all reasonable complexity models such as $Q$, $D$, $R$, $R_0$, $R_1$, $Q_1$, $Q_0$, $Q_E$, $C$, $RC$, $bs$, $\deg$, and $\widetilde{\deg}$.*

*Proof.* Consider restricting $f_{\mathrm{CS}(\phi)}$ to a promise that guarantees that all cheat sheets of the input are identical. An input $z$ to this promise problem evaluates to 1 if and only if the $\phi$ inputs from the first cheat sheet all evaluate to 1. We add the additional promise that this is true for all but the last input to $\phi$. The value of an input $z$ to this function then becomes exactly $\phi$ evaluated on a certain portion of the input. The complexity measures mentioned do not increase when the function is restricted to a promise. Thus the complexity of $f_{\mathrm{CS}(\phi)}$ under these measures is at least that of $\phi$. □

**Lemma 21.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$, and let $\phi$ be a certifying function for $f$. Then $D(f_{\mathrm{CS}(\phi)}) = \Omega(D(f))$.*

*Proof.* We describe an adversary strategy that can be used to force a deterministic algorithm to make $\Omega(D(f))$ queries. Let $x^1, x^2, \ldots, x^c$ represent the $c$ inputs to $f$. Whenever the algorithm queries a bit not in $x^1, x^2, \ldots, x^c$, the adversary returns 0. Whenever the algorithm queries a bit in $x^i$, the adversary uses the adversarial strategy for $f$ on that input; that is, it returns answers that are consistent with the promise of $f$, but that force any deterministic algorithm to use $D(f)$ queries to discern $f(x^i)$.

Now if a deterministic algorithm uses fewer than $D(f)$ queries on an input to $f_{\mathrm{CS}(\phi)}$, then there must be many cheat sheets in the input that it did not query at all. In addition, such an

algorithm cannot determine the value of $f(x^i)$ for any $i$. It follows that this algorithm could not have discerned the value of $f_{\mathrm{CS}(\phi)}$ on the given input, and thus $D(f_{\mathrm{CS}(\phi)}) \geq D(f)$. □

**Lemma 22.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$, and let $\phi$ be a certifying function for $f$. Then $R_0(f_{\mathrm{CS}(\phi)}) = \Omega(R_0(f))$.*

*Proof.* Let $A$ be a zero-error randomized algorithm for $f_{\mathrm{CS}(\phi)}$. Then $A$ must always find a certificate for $f_{\mathrm{CS}(\phi)}$. Consider running $A$ on an input $z$ consisting of $c$ 0-inputs to $f$, together with an all-zeros array. Certifying such an input requires presenting 0-certificates for some of the $c$ inputs to $f$, and presenting at least one bit from each cheat sheet whose index has not been disproven by the 0-certificates of the $f$ inputs.

Now, since there are at least $D(f)^{10}$ cheat sheets, only a small fraction can be provided in a certificate of size at most $O(R_0(f))$. It follows that the algorithm $A$ must find certificates to at least half of the $c$ 0-inputs to $f$ with high probability.

We can now turn $A$ into a new algorithm $B$ that finds a certificate in a given 0-input to $f$ sampled from the hard distribution $\mathcal{D}^0$ over the 0-inputs of $f$. Given such an input $x$, the algorithm $B$ will generate $c-1$ additional inputs from $\mathcal{D}^0$, shuffle them together, and add an all-zero array. $B$ will then feed this input to $A$. We know that with high probability, $A$ will find a certificate for at least half the inputs to $f$, so it must find a certificate for $x$ with probability close to $1/2$ (say, probability at least $1/3$). If it does not return a certificate, $B$ can repeat this process. The expected running time of $B$ is at most 3 times that of $A$.

Similarly, if we feed $A$ an input consisting of $c$ 1-inputs to $f$ together with an all-zero array, we can use a similar argument to construct an algorithm $C$ for finding 1-certificates in inputs sampled from $\mathcal{D}^1$ (the hard distribution over 1-inputs for $f$). We can then alternate running steps of $B$ and steps of $C$ to get an algorithm that finds a certificate in an input sampled from either $\mathcal{D}^0$ or $\mathcal{D}^1$, which uses an expected number of queries that is at most 6 times that of $A$.

This last algorithm evaluates $f$ on inputs sampled from the hard distribution for $f$, and therefore must have expected running time at least $R_0(f)$. It follows that the expected running time of $A$ is at least $R_0(f)/6$. □

The lower bounds for bounded-error randomized query complexity and bounded-error quantum query complexity are exactly as in Lemma 6 and Lemma 12.

### 6.4 Proofs of known results using cheat sheets

**Theorem 23** ([ABB+15])**.** *There exists a total function $f$ such that $R_0(f) = \widetilde{\Omega}(Q(f)^3)$.*

*Proof.* This proof is similar to that of Theorem 1, except with a different function $g$. Let $g : D \to \{0,1\}$, where $D \in \{0,1\}^n$, be a partial function that satisfies $Q(g) = 1$ and $R_0(g) = \Omega(n)$, such as the Deutsch–Jozsa problem [DJ92, CEMM98]. $R_0(g) = \Omega(n)$ follows from the fact that even certifying that the input is all zeros or all ones requires a certificate of linear size.

Let $h : \{0,1\}^m \to \{0,1\}$ be the AND-OR function on $m$ bits. Let $f = g \circ h$ and let $\phi$ be a certifying function that requires the obvious certificate for $f$ (as used in the proof of Theorem 1). Then we have $Q(\phi) = O(n + \sqrt{n}m^{1/4})$ and $Q(f) = Q(g \circ h) = O(\sqrt{m})$. From Theorem 5 we have that $R_0(f) = R_0(g \circ h) = \widetilde{\Omega}(nm)$. Using Theorem 16, we have $Q(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(Q(f) + Q(\phi)) = \widetilde{O}(\sqrt{m} + n + \sqrt{n}m^{1/4})$ and $R(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(R_0(f)) = \widetilde{\Omega}(nm)$. Plugging in $m = n^2$, we get $Q(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(n)$ and $R(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(n^3)$. □

**Theorem 24** ([ABB+15])**.** *There exists a total function $f$ such that $R(f) = \widetilde{\Omega}(Q_E(f)^{3/2})$ .*

*Proof.* Let $g : D \to \{0,1\}$, where $D = \{x \in \{0,1\}^n : |x| = 0 \text{ or } |x| = 1\}$, be the partial function that evaluates to 0 if the input is the all-zeros string and evaluates to 1 if the input string has Hamming weight 1. This function satisfies $R(g) = \Omega(n)$, since the block sensitivity of the 0-input is linear and $Q_E(g) = O(\sqrt{n})$, since Grover's algorithm can be made exact in this case [BHMT02].

Let $h : \{0,1\}^m \to \{0,1\}$ be the AND function on $m$ bits, which satisfies $R(h) = \Omega(m)$ and $Q_E(h) = O(m)$. Let $f = g \circ h$, and let $\phi$ be a certifying function for $f$. The certificate complexity of this function is $O(n)$ in the no case (by showing one 0-input to each of the $n$ AND gates), and is $O(m)$ in the yes case (by showing all the 1 inputs to an AND gates). We choose the certifying function $\phi$ that requires this certificate for $f$ and hence $Q_E(\phi) \leq D(\phi) = \widetilde{O}(n+m)$.

Now by composing quantum algorithms for $g$ and $h$ we have $Q_E(f) = O(m\sqrt{n})$, and by Theorem 5 we have $R(f) = \Omega(mn)$. Hence using Theorem 16, we have $Q_E(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(Q_E(f) + Q_E(\phi)) = \widetilde{O}(n+m+m\sqrt{n})$, and $R(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(mn)$. Choosing $m = \sqrt{n}$ gives $Q_E(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(n)$ and $R(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(n^{3/2})$. □

## Acknowledgements

We thank Ansis Rosmanis for comments on Appendix A and Mark Bun for discussions on approximate degree lower bounds. We thank Hardik Bansal for spotting an error in an earlier version of the proof of Theorem 5. We also thank Iordanis Kerenidis, Frédéric Magniez, and Ashwin Nayak for catching a bug in an earlier version of the proof of Lemma 6.

This work was partially funded by the ARO grant Contract Number W911NF-12-1-0486, as well as by an Alan T. Waterman Award from the National Science Foundation, under grant no. 1249349. This preprint is MIT-CTP #4730.

## References

[AA15] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC 2015)*, pages 307–316, 2015. [pp. 2, 4, 9]

[Aar06] Scott Aaronson. Quantum certificate complexity. *SIAM Journal on Computing*, 35(4):804–824, 2006. [p. 8]

[ABB+15] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *arXiv preprint* arXiv:1506.04719, 2015. [pp. 2, 3, 4, 5, 7, 24, 30]

[AdW14] Andris Ambainis and Ronald de Wolf. How low can approximate degree and quantum query complexity be for total boolean functions? *Comput. Complex.*, 23(2):305–322, June 2014. [p. 3]

[Amb03] Andris Ambainis. Polynomial degree vs. quantum query complexity. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 230–239, 2003. [p. 3]

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007. [p. 2]

[Amb13] Andris Ambainis. Superlinear advantage for exact quantum algorithms. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC 2013)*, pages 891–900, 2013. [p. 5]

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, July 2004. [p. 3]

[BBBV97] 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. [pp. 2, 6, 17]

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. [pp. 2, 3, 8]

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21 – 43, 2002. [pp. 7, 23]

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum computation and information*, volume 305 of *Contemporary Mathematics*, pages 53–74. AMS, 2002. [p. 25]

[BNRdW07] Harry Buhrman, Ilan Newman, Hein Rohrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory of Computing Systems*, 40(4):379–395, 2007. [pp. 20, 23]

[BŠ13] Aleksandrs Belovs and Robert Špalek. Adversary lower bound for the k-sum problem. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 323–328, 2013. [pp. 6, 14, 28]

[BT15] Mark Bun and Justin Thaler. Hardness amplification and the approximate degree of constant-depth circuits. In *Automata, Languages, and Programming*, volume 9134 of *Lecture Notes in Computer Science*, pages 268–280. 2015. [p. 5]

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998. [p. 24]

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 439(1907):553–558, 1992. [p. 24]

[GJPW15] Mika Göös, T.S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. *Electronic Colloquium on Computational Complexity (ECCC) TR15-169*, 2015. [pp. 4, 5]

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *Electronic Colloquium on Computational Complexity (ECCC) TR15-050*, 2015. [pp. 2, 3, 4, 5, 9]

[Gro96] 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. [p. 2]

[GSS13] 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. [pp. 5, 9]

[HLŠ07] 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. [pp. 3, 14]

[HMdW03] Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 291–299. 2003. [p. 10]

[JK10] 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. [p. 10]

[Kim12] Shelby Kimmel. Quantum adversary (upper) bound. In *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 557–568. 2012. [p. 14]

[KT13] Raghav Kulkarni and Avishay Tal. On fractional block sensitivity. *Electronic Colloquium on Computational Complexity (ECCC) TR13-168*, 2013. [p. 8]

[LG06] François Le Gall. Exponential separation of quantum and classical online space complexity. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '06, pages 67–73, New York, NY, USA, 2006. ACM. [p. 3]

[LMR+11] 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. [pp. 10, 14, 18, 19]

[LR13] Troy Lee and Jérémie Roland. A strong direct product theorem for quantum query complexity. *computational complexity*, 22(2):429–462, 2013. [pp. 6, 16]

[Mid04] Gatis Midrijanis. Exact quantum query complexity for total Boolean functions. *arXiv preprint* `arXiv:quant-ph/0403168`, 2004. [p. 8]

[Nis91] Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. [pp. 2, 8]

[NS95] Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 15(4):557–565, 1995. [p. 3]

[NW95] Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995. [p. 5]

[Rei11] 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. [pp. 10, 14]

[San95] Miklos Santha. On the Monte Carlo Boolean decision tree complexity of read-once formulae. *Random Structures & Algorithms*, 6(1):75–87, 1995. [p. 5]

[She13] Alexander A. Sherstov. Approximating the AND-OR tree. *Theory of Computing*, 9(20):653–663, 2013. [p. 5]

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. [p. 2]

[Sim97] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. [pp. 4, 9]

[ŠS06] Robert Špalek and Mario Szegedy. All quantum adversary methods are equivalent. *Theory of Computing*, 2(1):1–18, 2006. [p. 3]

[SW86] Michael Saks and Avi Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS 1986)*, pages 29–38, 1986. [pp. 2, 5]

[Tal13] 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. [p. 8]

# A Quantum query complexity of $k$-sum

Belovs and Špalek [BŠ13] proved a lower bound of $\Omega(n^{k/(k+1)})$ on the quantum query complexity of $k$-SUM for constant $k$, as long as the alphabet is large enough. However, in their result, Belovs and Špalek do not keep track of the dependence on $k$. We trace their proof to determine this dependence, ultimately proving the following theorem.

**Theorem 25.** *For the function $k$-SUM : $[q]^n \to \{0,1\}$, if $|q| \geq 2\binom{n}{k}$, we have*

$$Q(k\text{-SUM}) = \Omega(n^{k/(k+1)}/\sqrt{k}), \tag{16}$$

*where the constant in the $\Omega$ is independent of $k$.*

*Proof.* We proceed by tracing Belovs and Špalek's proof, which can be found in section 3 of [BŠ13], to extract an unsimplified lower bound. We then tune some parameters to determine the best dependence on $k$.

The proof in [BŠ13] proceeds by the generalized adversary method. Belovs and Špalek construct an adversary matrix $\Gamma$, which depends on parameters $\alpha_m$ for $m = 0, 1, \ldots, n-k$. This construction can be found in Section 3.1 of their paper. They then show a lower bound on $\|\Gamma\|$ and an upper bound on $\|\Gamma \circ \Delta_1\|$ in terms of the parameters $\alpha_m$ (the upper bounds on $\|\Gamma \circ \Delta_i\|$ for $i \neq 1$ follow by symmetry). Finally, they pick the values for the parameters that give the best lower bound.

A lower bound on $\|\Gamma\|$ without $\Omega$ notation, which therefore does not ignore factors of $k$, can be found in Section 3.6 of their paper. In that section, they show

$$\|\Gamma\| \geq \alpha_0 \sqrt{\binom{n}{k}\left(1 - \frac{1}{q}\binom{n}{k}\right)}, \tag{17}$$

where $q$ is the alphabet size. When $q \geq \frac{4}{3}\binom{n}{k}$, this gives

$$\|\Gamma\| \geq \frac{\alpha_0}{2}\sqrt{\binom{n}{k}}. \tag{18}$$

For the upper bound on $\|\Gamma \circ \Delta_1\|$, it turns out to be more convenient to switch to a different matrix, $\tilde{\Gamma}$, which satisfies $\|\Gamma \circ \Delta_1\| \leq \|\tilde{\Gamma} \circ \Delta_1\|$. This is explained in the beginning of Section 3 of their paper and in Section 3.1. To upper bound $\|\tilde{\Gamma} \circ \Delta_1\|$, in Section 3.3 the authors switch to yet another matrix, $\tilde{\Gamma}_1$, with the property that $\tilde{\Gamma}_1 \circ \Delta_1 = \tilde{\Gamma} \circ \Delta_1$ and that $\|\tilde{\Gamma}_1 \circ \Delta_1\| \leq 2\|\tilde{\Gamma}_1\|$. This gives $\|\Gamma \circ \Delta_1\| \leq 2\|\tilde{\Gamma}_1\|$, so it suffices to upper bound $\|\tilde{\Gamma}_1\|$.

In Section 3.4, the authors express $\|\tilde{\Gamma}_1\|^2$ as the norm of a sum of matrices, with the sum ranging over a set $\mathcal{S}$. They split $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$; by the triangle inequality, it then suffices to separately upper bound the norm of the sum over $\mathcal{S}_1$ and the norm of the sum over $\mathcal{S}_2$. The former is upper bounded by $\max_m \alpha_m^2 \binom{m+k-1}{k-1}$, and the latter by $k\binom{n-1}{k} \max_m (\alpha_m - \alpha_{m+1})^2$. We therefore conclude that

$$\|\Gamma \circ \Delta_1\|^2 \leq 4 \max_m \alpha_m^2 \binom{m+k-1}{k-1} + 4k\binom{n-1}{k} \max_m (\alpha_m - \alpha_{m+1})^2. \tag{19}$$

It remains to pick values for $\alpha_m$ to optimize the resulting adversary bound. There are no restrictions on the $\alpha_m$ parameters. To maximize the adversary bound, we want $\alpha_0$ to be large, consecutive $\alpha$'s to be close (to minimize $\max_m (\alpha_m - \alpha_{m+1})^2$), and $\alpha_m$ for large $m$ to be small (to minimize $\max_m \alpha_m^2 \binom{m+k-1}{k-1}$). We choose to make the $\alpha_m$ parameters decrease to 0 in an arithmetic sequence, with $\alpha_m - \alpha_{m+1} = c > 0$ for all $m \leq \alpha_0/c$, and $\alpha_m = 0$ for all $m \geq \alpha_0/c$. Let $\beta = \alpha_0/c$. Then $\alpha_m = \alpha_0 - cm$ for $m \leq \beta$.

We can write the final adversary bound as follows:

$$\mathrm{Adv}^{\pm}(k\text{-SUM})^2 = \Omega\left(\frac{\alpha_0^2 \binom{n}{k}}{\max_{m \leq \beta} (\alpha_0 - cm)^2 \binom{m+k-1}{k-1} + k\binom{n-1}{k} c^2}\right) \tag{20}$$

$$= \Omega\left(\min\left\{\frac{\binom{n}{k}}{\max_{m \leq \beta} (1 - m/\beta)^2 \binom{m+k-1}{k-1}}, \frac{\binom{n}{k}}{k\binom{n-1}{k}} \beta^2\right\}\right) \tag{21}$$

The second term in the minimization simplifies to $\frac{n}{k(n-k)}\beta^2$, which is $\Omega(\beta^2/k)$. To deal with the first term, we use the well-known inequalities $(n/k)^k \leq \binom{n}{k} \leq (en/k)^k$, which hold for all $n$ and $k$. This gives

$$\mathrm{Adv}^{\pm}(k\text{-SUM})^2 = \Omega\left(\min\left\{\frac{n^k}{(k-1)\max_{m \leq \beta} (1 - m/\beta)^2 (e(m+k-1))^{k-1}} \left(\frac{k-1}{k}\right)^k, \frac{\beta^2}{k}\right\}\right) \tag{22}$$

$$= \Omega\left(\min\left\{\frac{n^k}{k \max_{m \leq \beta} (1 - m/\beta)^2 (e(m+k))^{k-1}}, \frac{\beta^2}{k}\right\}\right). \tag{23}$$

We can solve the maximization in the denominator using calculus. The unique maximum occurs at $m = \beta - \left(\frac{2}{k+1}\right)(\beta - k)$. Substituting this in and simplifying, we get

$$\mathrm{Adv}^{\pm}(k\text{-SUM})^2 = \Omega\left(\min\left\{\frac{k\beta}{(1-k/\beta)^2}\left(\frac{n}{e(\beta+3)}\right)^k, \frac{\beta^2}{k}\right\}\right) = \Omega\left(\min\left\{k\beta\left(\frac{n}{3\beta}\right)^k, \frac{\beta^2}{k}\right\}\right), \tag{24}$$

where for the last equality we assumed $\beta \geq 2k$ and $\beta \geq 3e/(3-e)$.

Finally, we set $\beta = (1/3)n^{k/(k+1)}$. This gives

$$\mathrm{Adv}^{\pm}(k\text{-SUM})^2 = \Omega\left(\min\left\{kn^{k/(k+1)}\left(\frac{n}{n^{k/(k+1)}}\right)^k, \frac{n^{2k/(k+1)}}{k}\right\}\right) = \Omega(n^{2k/(k+1)}/k), \tag{25}$$

so

$$Q(k\text{-SUM}) = \Omega(\mathrm{Adv}^{\pm}(k\text{-SUM})) = \Omega\left(n^{k/(k+1)}\Big/\sqrt{k}\right). \tag{26}$$

Note that we assumed $\beta \geq 2k$. Since $\beta = n^{k/(k+1)}/3$, we need $n^{k/(k+1)} \geq 6k$, or $n \geq (6k)^{1+1/k}$. Since $(6k)^{1/k} = \exp(\ln(6k)/k) = 1+\Theta(\log(k)/k)$, it suffices to have $n \geq 6k+\omega(\log k)$. In particular, this bound works as long as we have $k \leq n/10$. When $k \geq n/10$, we can directly prove a lower bound of $\Omega(\sqrt{n})$ by a reduction from Grover search; this means there are no restrictions on the size of $n$ and $k$ in this result. □

# B Measures that behave curiously with cheat sheets

In this appendix we show that $R_1$ and $Q_1$ can behave strangely on cheat sheet functions, potentially decreasing from $R_1(f)$ to $R(f)$ and from $Q_1(f)$ to $Q(f)$.

**Theorem 26.** *Let $f : D \to \{0,1\}$ be a partial function, where $D \subseteq [M]^n$, and let $\phi$ be a certifying function for $f$. Then*

$$R_1(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(R(f) + R_0(\phi)) \qquad \text{and} \qquad Q_1(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(Q(f) + Q_0(\phi)). \tag{27}$$

*Proof.* We show a randomized (and quantum) algorithm for $f_{\mathrm{CS}(\phi)}$ that uses the required number of queries and finds a 1-certificate with constant probability. Such an algorithm could be made to have one-sided error by outputting 1 only if it finds a 1-certificate.

The algorithm works as follows. First, use the randomized (resp. quantum) algorithm on the $c$ inputs to $f$ (with some amplification), to determine the number $\ell$ of the correct cheat sheet with high probability (assuming the promises of $f$ all hold). Next, go to the $\ell^{\text{th}}$ cheat sheet, and use a zero-error algorithm to evaluate $\phi(x_i, y_i)$ for $i = 1, 2, \ldots, c$, where $x_i$ are the inputs to $f$ and $y_i$ are the cheat sheet strings. This finds certificates for each input to $\phi$.

Now, if the value of the function is 1 on the given input, then with constant probability, all the $c$ certificates found should be 1-certificates for $\phi$. These 1-certificates certify the value of the inputs to $f$. Therefore, taken together, they constitute a valid 1-certificate for $f_{\mathrm{CS}(\phi)}$. It follows that this algorithm uses $\widetilde{O}(R(f)+R_0(\phi))$ randomized queries and finds a 1-certificate with constant probability. The result for quantum algorithms follows similarly. □

We now show that it is possible for $R_1(f_{\mathrm{CS}(\phi)})$ to be much smaller than $R_1(f)$, unlike some of the other measures studied. Intuitively, this is because 1-inputs of $f_{\mathrm{CS}(\phi)}$ contains a cheat sheet with a certificate and hence even if the algorithm is not sure of its answer before finding the cheat sheet, the cheat sheet may convince the algorithm of its answer.

**Theorem 27.** *There is a total function $f : \{0,1\}^n \to \{0,1\}$ and a certifying function $\phi$ for $f$ such that $R_1(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(\sqrt{R_1(f)})$.*

*Proof.* We can use Theorem 26 to construct an explicit function $f$ and a certifying function $\phi$ for $f$ such that $R_1(f_{\mathrm{CS}(\phi)})$ is smaller than $R_1(f)$. The construction is as follows. In [ABB+15], a Boolean function was constructed that has $R_0 = \widetilde{\Omega}(n^2)$ and $R_1 = \widetilde{O}(n)$. This function has

certificate complexity roughly equal to $n$. Now, by taking the XOR of this function with an additional bit, we get a function $f$ for which $R_1$ is roughly $n^2$ but $R$ and $C$ are still roughly $n$.

We define $\phi$ to be a certifying function that simply checks if $y$ provides pointers to a certificate for $f$. Then $R_0(\phi) = \widetilde{O}(n)$. It follows that $R_1(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(n)$, while $R_1(f) = \widetilde{\Omega}(n^2)$. □

Lastly we show that even zero-error randomized query complexity behaves curiously with cheat sheets. We might have expected that the obvious upper bound $R_0(f_{\mathrm{CS}(\phi)}) = \widetilde{O}(R_0(f) + R_0(\phi))$ holds, but in fact it does not. The reason is subtle and relates to the behavior of the zero-error algorithm on inputs outside of the domain $D$. For a partial function $f : D \to \{0,1\}$, a zero-error algorithm's behavior is not constrained on inputs outside the domain $D$. The algorithm could, for example, run forever on such inputs. The obvious algorithm, which simply runs the zero-error algorithm for $f$ on all $c$ inputs to $f$ now fails to output any answer on 0-inputs to $f_{\mathrm{CS}(\phi)}$ in which the inputs to $f$ do not lie in $D$. We exploit this observation to prove the following counterexample.

**Theorem 28.** *There is a partial function $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$, and a certifying function $\phi$ for $f$ such that $R_0(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(R_0(f)^{3/2} + R_0(\phi)^{3/2})$.*

*Proof.* Let $g : \{0,1\}^{4m} \to \{0,1\}$ be the partial function defined as follows. On input $(x, y)$ with $x, y \in \{0,1\}^{2m}$, define $g(x, y) = 0$ if the Hamming weight of $x$ is $m$ and the Hamming weight of $y$ is 0, and define $g(x, y) = 1$ if the Hamming weight of $y$ is $m$ and the Hamming weight of $x$ is 0. The promise of $g$ is that one of these two conditions holds for every input $(x, y)$.

It is easy to see that $R_0(g) = O(1)$. Let $f$ be the composition of $g$ with an $m$ by $m$ AND-OR, and let $n = 4m^3$. Then $R_0(f) = O(m^2) = O(n^{2/3})$. Let $\phi$ be a certifying function for $f$ that takes an input to $f$ and an additional string, and asserts that the latter provides pointers to a certificate for each AND-OR instance in the input to $f$. This assertion can be verified (or refuted) in $\widetilde{O}(m^2)$ deterministic queries, so $R_0(\phi) \leq D(\phi) = \widetilde{O}(m^2)$.

We now show that $R_0(f_{\mathrm{CS}(\phi)}) = \widetilde{\Omega}(m^3)$. Consider an input consisting of $c$ inputs to $f$, each of which has all its AND-OR instances evaluate to 0. Note that these inputs to $f$ do not satisfy the promise of $f$. We attach a blank cheat sheet array after these $c$ inputs. By definition of $f_{\mathrm{CS}(\phi)}$, the resulting input is therefore a 0-input to $f_{\mathrm{CS}(\phi)}$. However, any 0-certificate of it (of reasonable size) must "partially certify" at least half of the $c$ inputs to $f$; that is, for at least half the inputs to $f$, it must either prove the input is not a 0-input or prove it is not a 1-input. The only way to do this is to display $(c/2)m$ 0-certificates for AND-OR instances.

This means we have an algorithm that takes in $4cm$ zero-inputs to AND-OR, and finds a certificate for at least $cm/8$ of them using $R_0(f_{\mathrm{CS}(\phi)})$ expected queries. It follows that given one input from the hard distribution over 0-inputs to $f$, we can find a certificate for it by generating $4cm - 1$ additional inputs from this distribution, mixing them together, and running the $R_0(f_{\mathrm{CS}(\phi)})$ algorithm. This finds a certificate with probability at least $1/8$, and uses $R_0(f_{\mathrm{CS}(\phi)})/cm$ expected queries. By repeating the algorithm if necessary, we find a certificate using $8R_0(f_{\mathrm{CS}(\phi)})/cm$ expected queries.

We construct a function $f'$ is the same as $f$ except with NOT-AND-OR instead of AND-OR. Repeating the argument provides a randomized algorithm that finds a 1-certificate for AND-OR using $8R_0(f'_{CS(\phi')})/cm$ expected queries. By running both algorithms in parallel, we get

$$R_0(\text{AND-OR}) \leq 8R_0(f_{\mathrm{CS}(\phi)})/cm + 8R_0(f'_{\mathrm{CS}(\phi')})/cm. \tag{28}$$

Since $R_0(\text{AND-OR}) = \Omega(m^2)$, it follows that either $R_0(f_{\mathrm{CS}(\phi)}) = \Omega(m^3)$ or $R_0(f'_{\mathrm{CS}(\phi')}) = \Omega(m^3)$. The desired result follows. □