Quantum advantage and lower bounds in parallel query complexity

Joseph Carolan[*], Amin Shiraz Gilani[†], Mahathi Vempati[‡]

Department of Computer Science, University of Maryland
Institute for Advanced Computer Studies, University of Maryland
Joint Center for Quantum Information and Computer Science, University of Maryland

Abstract

It is well known that quantum, randomized and deterministic (sequential) query complexities are polynomially related for total boolean functions. We find that significantly larger separations between the parallel generalizations of these measures are possible. In particular,

(1) We employ the cheatsheet framework to obtain an unbounded parallel quantum query advantage over its randomized analogue for a total function, falsifying a conjecture of [Jeffery et al. 2017].
(2) We strengthen (1) by constructing a total function which exhibits an unbounded parallel quantum query advantage despite having no sequential advantage, suggesting that genuine quantum advantage could occur entirely due to parallelism.
(3) We construct a total function that exhibits a polynomial separation between 2-round quantum and randomized query complexities, contrasting a result of [Montanaro. 2010] that there is at most a constant separation for 1-round (nonadaptive) algorithms.
(4) We develop a new technique for deriving parallel quantum lower bounds from sequential upper bounds. We employ this technique to give lower bounds for Boolean symmetric functions and read-once formulas, ruling out large parallel query advantages for them.

We also provide separations between randomized and deterministic parallel query complexities analogous to items (1)-(3).

[*]jcarolan@umd.edu
[†]asgilani@umd.edu
[‡]mahathi@umd.edu

Contents

1 Introduction

Quantum query complexity is a widely studied model for understanding the capabilities and limitations of quantum computers. Many of the important quantum algorithms and quantum-classical separation results are expressed in this model. For instance, the quantum period finding algorithm, a major ingredient in Shor's factoring algorithm [Sho94], was first developed in the query model. There are many other examples of query problems for which quantum algorithms provably achieve exponential query advantage over their classical counterparts [Sim97, CCD+03].

However, these problems have partial domains, meaning they are not defined on most inputs. For functions whose domain includes all inputs (total functions), it is known that the quantum, randomized and deterministic query complexities are all polynomially related [BBC+98, ABDK+21]. Grover's seminal algorithm [Gro96] for unstructured search achieves a quadratic speed-up over classical algorithms, and Ambainis et al. [ABB+17] and Aaronson et al. [ABK16] construct functions that separate quantum and deterministic query complexities by a 4th power, and quantum and randomized query complexities by a 3rd power respectively.* These separations cannot be significantly improved in the sequential query model [BBC+98, ABDK+21].

A natural question is whether these algorithmic limitations are robust against generalizations of sequential query complexity. In this work, we consider a parallel model where an algorithm is allowed to make many queries at each step, as introduced in [JMW17]. In particular, the p-parallel quantum query complexity, denoted by $\mathsf{Q}^{p\|}$, of a function is the minimum number of steps needed to compute it with bounded error, if at each step the algorithm is allowed to make p non-adaptive quantum queries.† We will also refer to this quantity by *p-query depth*, *p-query rounds* and *p-query layers* interchangeably. Notice that, for any function f and any parallelism p, $\mathsf{Q}(f)/p \leq \mathsf{Q}^{p\|}(f) \leq \mathsf{Q}(f)$ since any q-round p-parallel algorithm can be simulated by a pq-query sequential algorithm, and any q-query sequential algorithm can be simulated by a q-round p-parallel algorithm.‡

This model is motivated by the fact that fault tolerant quantum computers must be inherently parallel: a large scale quantum computer which cannot do many gates at once cannot correct local errors faster than they occur. Therefore, quantum computers will necessarily be able to perform operations in parallel, meaning one should design algorithms to take maximum advantage of this capability. Further, near term quantum computations are limited to low depth due to short decoherence times. These factors make the parallel query model a more natural abstraction of real devices than a purely sequential model.

Prior work addressing parallel query complexity has indicated that quantum advantage tends to disappear as parallelism increases. Zalka [Zal99] showed the parallel complexity for searching an unstructured list of size N is $\Theta\left(\sqrt{N/p}\right)$. This was generalized by Grover and Radhakrishnan [GR04], who proved that $\tilde{\Theta}\left(\sqrt{Nt/(p \cdot \min\{t,p\})}\right)$ p-parallel queries are necessary and sufficient to find t marked elements in a list of size N. Along with providing a systematic study of parallel quantum query complexity, Jeffery, Magniez and de Wolf [JMW17] showed that the p-parallel quantum query complexity of k-SUM is $\tilde{\Theta}\left((N/p)^{k/k+1}\right)$.

Furthermore, [JMW17] introduce the following conjecture:

*[ABK16] originally showed a 2.5th power separation, which was boosted to a 3rd power via tight lower bounds on k-fold Forrelation shown in [BS21, SSW21].

†We define the deterministic parallel query complexity $\mathsf{D}^{p\|}$ and randomized bounded error parallel query complexity $\mathsf{R}^{p\|}$ similarly.

‡Similar bounds hold for $\mathsf{R}^{p\|}$ and $\mathsf{D}^{p\|}$.

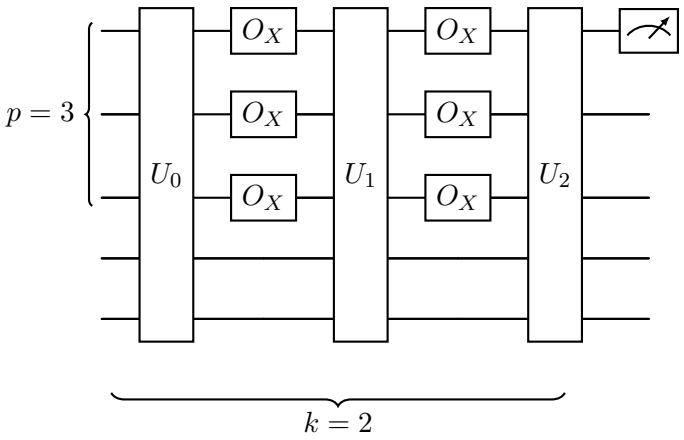

Figure 1: A parallel quantum algorithm with $k = 2$ adaptive steps and $p = 3$ queries per step; the last two wires are for workspace. In general, p denotes the number of non-adaptive queries per step, and k denotes the number of steps. The measure $Q^{p\|}$ denotes the minimal k needed to compute a function for a fixed p, while the measure $Q^{k\perp}$ denotes the minimal p needed to compute a function for a fixed k.

Conjecture 1 ([JMW17])**.** *For any total function $f : \{0,1\}^N \to \{0,1\}$ and any p, there is at most a polynomial quantum advantage for p-parallel query algorithms. That is,*

$$\mathsf{D}^{p\|}(f) = \mathsf{poly}(\mathsf{Q}^{p\|}(f)).$$

In fact, [JMW17] proved this conjecture for all p polynomially smaller than the block sensitivity (bs, defined in Section 3) of f, generalizing a result of Beals et al. [BBC+98].

The notion of p-parallel query complexity captures the minimum "depth" needed to compute some function for a given "width" (see Figure 1). An alternative complexity measure relevant to parallelism is the minimum "width" needed to compute some function for a fixed "depth", which we call the k-adaptive query complexity, and denote by $\mathsf{Q}^{k\perp}(f)$ (respectively $\mathsf{R}^{k\perp}(f)$, $\mathsf{D}^{k\perp}(f)$). Precisely, we define k-adaptive query complexity as the minimum p such that there is a p-parallel quantum (respectively randomized, deterministic) algorithm which makes k many p-parallel queries and computes f. This complexity measure is not widely studied for total functions. However, a result of Montanaro [Mon10] fully characterizes the relevant 1-adaptive (or non-adaptive) query complexities, up to a factor of 2: for any total $f : \{0,1\}^N \to \{0,1\}$, $\mathsf{Q}^{1\perp}(f) \geq \mathsf{D}^{1\perp}(f))/2$. In the context of partial functions, a result by Girish et al. [GSTW23] characterized the maximal separation between $\mathsf{Q}^{k\perp}$ and $\mathsf{Q}^{k+1\perp}$.

1.1 Main results

1.1.1 Unbounded parallel separations for total functions

Our first result involves adapting the cheatsheet framework of Aaronson et al. [ABK16] to the parallel query setting, and using it to exhibit total function separations between parallel query complexity measures. In particular, we show that the canonical cheatsheet function (see Definition 21) exhibits unbounded separation between quantum and randomized parallel query complexities, falsifying Conjecture 1. Further, we observe that this function demonstrates that 3-adaptive quantum algorithms can be more efficient in terms of total queries than even fully sequential randomized algorithms. More formally, we prove the following theorem.

Theorem 2 (Restatement of Theorem 30)**.** *There exists a (total) Boolean function $h : \{0,1\}^N \to \{0,1\}$ such that for some $p = \tilde{O}(\mathsf{bs}(h))$ and some constants $\epsilon, \delta > 0$, we have*

(i) $\mathsf{R}^{p\|}(h) = \tilde{\Omega}(N^{\epsilon})$, *(ii)* $\mathsf{Q}^{p\|}(h) \leq 3$, *(iii)* $\mathsf{Q}^{3\perp}(h) = \tilde{O}\left(\mathsf{R}(h)^{1-\delta}\right)$.

It is worth noting that the large separation in parallel query complexity occurs when the parallelism almost exactly equals the block sensitivity $\mathsf{bs}(f)$, contrasting a result of Jeffery et al ([JMW17], Theorem 12) which eliminates this possibility for $p = O(\mathsf{bs}(f)^{1-\epsilon})$ for any constant $\epsilon > 0$. Furthermore, the separation between $\mathsf{Q}^{3\perp}(f)$ and $\mathsf{R}(f)$ contrasts Montanaro's result ([Mon10], Theorem 1) that for total functions, 1-adaptive quantum algorithms cannot be more than twice as efficient as the trivial 1-adaptive deterministic algorithm.

We also show a total function that achieves a similar separation between randomized and deterministic parallel query complexities (see Theorem 27). We do this by modifying the canonical cheatsheet function to instead embed a partial function with a large randomized-deterministic separation.

It is worth noting that this result is also sufficient to falsify Conjecture 1.

These results provide a recipe for achieving exponential parallel query advantages (from quantumness or randomness) for total functions by amplifying a polynomial sequential advantage for such functions. However, it is unclear whether an advantage can originate entirely from a quantum or randomized algorithm's ability to utilize parallelism more effectively than a randomized or deterministic algorithms. In other words, is there a function which has no sequential quantum query advantage, yet a large parallel quantum query advantage? We answer this question in the affirmative, even for total functions.

1.1.2 Unbounded genuine parallel separations

Our starting point is a partial function with the desired property. That is, we construct a partial function with (almost) equal randomized and quantum sequential query complexities, that does not admit any parallel advantage for randomized algorithms, yet admits near-maximal parallel advantage for quantum algorithms. In particular, we show the following theorem.

Theorem 3 (Restatement of Theorem 46)**.** *There exists a (partial) Boolean function $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subset \{0,1\}^N$ such that for some $p = O(\mathsf{Q}(f))$ and some constant $\epsilon > 0$, we have*

(i) $\mathsf{R}(f) = \tilde{\Theta}(N^{\epsilon})$, *(ii)* $\mathsf{Q}(f) = \tilde{\Omega}(\mathsf{R}(f))$, *(iii)* $\mathsf{R}^{p\|}(f) = \tilde{\Omega}(\mathsf{R}(f))$, *(iv)* $\mathsf{Q}^{p\|}(f) = 1$.

Note that the function f in the above theorem perfectly parallelizes quantumly but does not parallelize at all classically. We discuss the construction of f in more detail in the technical overview. We totalize f using the cheatsheet framework to give a total function with no sequential quantum advantage but unbounded parallel quantum advantage, giving rise to the following theorem.

Theorem 4 (Restatement of Theorem 48)**.** *There exists a (total) Boolean function $h : \{0,1\}^N \to \{0,1\}$ such that for some p and some constants $\epsilon, \delta > 0$ (with $\delta < \epsilon$), we have*

(i) $\mathsf{R}(h) = \tilde{\Theta}(N^{\epsilon})$, *(ii)* $\mathsf{Q}(h) = \tilde{\Omega}(\mathsf{R}(h))$, *(iii)* $\mathsf{R}^{p\|}(h) = \tilde{\Omega}(N^{\delta})$, *(iv)* $\mathsf{Q}^{p\|}(h) \leq 3$.

We also obtain analogous separations between the randomized and deterministic query complexities using the same techniques (see Theorems 41 and 43).

1.1.3 Separations with two layers of adaptivity

We earlier noted (in Theorem 2) that for some total function h, $\mathsf{Q}^{3\perp}(h)$ can be polynomially smaller than $\mathsf{R}^{3\perp}(h)$ (and even $\mathsf{R}(h)$), however $\mathsf{Q}^{1\perp}(h) = \Theta(\mathsf{D}^{1\perp}(h))$ for all total functions h. A natural

question to ask is whether it is possible for $\mathsf{Q}^{2\perp}(h)$ to be much smaller than $\mathsf{R}^{2\perp}(h)$ for some total h? We answer this question in the affirmative by presenting a general framework that transforms an (at least) polynomial separation between $\mathsf{Q}^{1\perp}(f)$ and $\mathsf{R}^{1\perp}(f)$ for some partial f to a polynomial separation between $\mathsf{Q}^{2\perp}(h)$ and $\mathsf{R}^{2\perp}(h)$ for some total h constructed from f. Since there exists a partial function f (such as Forrelation defined in Problem 14) with $\mathsf{Q}(f) = 1$ (so $\mathsf{Q}^{1\perp}(f) = 1$) and $\mathsf{R}(f) = \Omega(N^\epsilon)$ (so $\mathsf{R}^{1\perp}(f) = \Omega(N^\epsilon)$) for some $\epsilon > 0$, we get a total function h exhibiting a polynomial separation between $\mathsf{Q}^{2\perp}(h)$ and $\mathsf{R}^{2\perp}(h)$. Precisely, we show the following theorem:

Theorem 5 (Restatement of Theorem 55)**.** *Let $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subseteq \{0,1\}^N$ be a (partial) function that satisfies $\mathsf{Q}^{1\perp}(f) = \tilde{O}(N^\epsilon)$ and $\mathsf{R}^{1\perp}(f) = \tilde{\Omega}(N^\delta)$ for some constants $\epsilon < \delta$. Then there exists a total $h : \{0,1\}^M \to \{0,1\}$ (with $M = \Theta(N^3)$) and constants $\epsilon' < \delta'$ such that $\mathsf{Q}^{2\perp}(h) = \tilde{O}(M^{\epsilon'})$ and $\mathsf{R}^{2\perp}(h) = \tilde{\Omega}(M^{\delta'})$.*

We also obtain a similar separation between $\mathsf{R}^{2\perp}$ and $\mathsf{D}^{2\perp}$ (see Theorem 52).

1.1.4 Quantum parallel lower bound framework and applications

Finally, we develop a new method for deriving parallel quantum query lower bounds from sequential quantum query upper bounds. It is simpler to reason about sequential algorithms than parallel algorithms, and upper bounds are often easier to give than lower bounds. Thus, our result allows reducing the hard problem of lower bounding parallel quantum query complexity to the potentially easier problem of upper bounding its sequential analogue. Our technique is based on the parallel spectral adversary method [JMW17], and is especially well suited to problems with optimal adversary matrices that only distinguish input pairs that differ at a single index. We also show that, for any total function f, the combinatorial parallel adversary method [GR04, Bur19] (see Theorem 18) fails to show a lower bound better than $\Omega\left(\sqrt{\left\lceil \frac{\mathsf{C}_0(f)}{p} \right\rceil \left\lceil \frac{\mathsf{C}_1(f)}{p} \right\rceil}\right)$ where C_b denotes the bth certificate complexity of f (see Theorem 58). Moreover, we show that our method surpasses this barrier and provides a better lower bound, for instance, to the And $\circ$ Or problem.

We let $\lambda(f)$ denote the spectral sensitivity of f, as defined in Theorem 60. Let $\mathcal{F}^{(f)}_{p-\text{res}}$ denote the set of all functions obtained from restricting f to p input bits, and fixing the rest. We call this a $p-$restriction of f.

Theorem 6 (Restatement of Theorem 61)**.** *For any total Boolean function $f : \{0,1\}^N \to \{0,1\}$, we have*

$$\mathsf{Q}^{p\|}(f) = \Omega\left(\lambda(f) \cdot \frac{1}{\max_{g \in \mathcal{F}^{(f)}_{p-res}} \lambda(g)}\right)$$

Recalling that $\lambda(f) = O(\mathsf{Q}(f))$, as shown by Aaronson et al. [ABDK+21], we can write this as $\mathsf{Q}^{p\|}(f) = \Omega\left(\lambda(f) \cdot (\max_{g \in \mathcal{F}^{(f)}_{p-\text{res}}} \mathsf{Q}(g))^{-1}\right)$, where we note that the second term can be lower bounded by giving a (sequential) quantum algorithm for any $p-$restriction of f. We apply Theorem 6 to the following classes of well-studied functions.

1. *Read-once formulas (*Section 7.3): While the combinatorial adversary method [Amb02] is sufficient to lower bound the sequential quantum query complexity of the two-layer And $\circ$ Or tree by $\Omega(\sqrt{N})$ [BS04], the best lower bound the parallel combinatorial adversary method can give for the Read-once formula problem $\mathsf{And}_{\sqrt{N}} \circ \mathsf{Or}_{\sqrt{N}}$ is $\Omega(\sqrt{N}/p)$ (see Corollary 59). Using Theorem 6, we show a $\Omega\left(\sqrt{N/p}\right)$ lower bound for any read-once formula (Theorem 63).

2. *Symmetric functions* (Section 7.4): From Theorem 6, we get the tight lower bound of $\Omega\left(\sqrt{Nt/p \cdot \min\{t, p\}}\right)$ where t is the largest hamming weight less than $N/2$ such that the function value either differs on inputs of hamming weight t and $t+1$ or differs on inputs of hamming weight $N-t$ and $N-t-1$ (see Theorem 64). This result recovers the combinatorial adversary lower bound implicit in [GR04] using an arguably simpler proof.

1.2 Open questions

A priori, it seems unintuitive that an exponential advantage in number of rounds is possible for total functions, and also seems like lower bounding a simple function like $\textsc{And} \circ \textsc{Or}$ in the parallel setting should be possible by standard techniques. Our results in Section 1.1.1 and Section 1.1.4 show that former is indeed true, and the latter is not the case and in fact requires "parallel-specific" thinking. Thus, one of our contributions is highlighting that there are still several fundamental query complexity questions unanswered in the parallel setting. We discuss a couple of these:

Limitations of parallel speedup. Can classical (randomized or deterministic) query algorithms simulate quantum query algorithms when allowed *both* a polynomial overhead in parallelism as well as number of rounds (as opposed to just a polynomial overhead in number of rounds as in Conjecture 1)? We conjecture that this is true:

Conjecture 7. *For all total functions $f : \{0,1\}^M \to \{0,1\}$ and parallelism p, we have*

$$\mathsf{D}^{\mathsf{poly}(p)\|}(f) = \mathsf{poly}(\mathsf{Q}^{p\|}(f)).$$

- When $p < \mathsf{bs}(f)^{1-\epsilon}$ for any $\epsilon > 0$, [JMW17] prove that this is true.
- When $p \geq \mathsf{bs}(f) = \Omega(M^{\epsilon})$ for any $\epsilon > 0$, clearly this is true. This is the case in our result as well, where the canonical cheat sheet function (Definition 21) used to show Theorem 2 has $\mathsf{bs}(f) = \tilde{\Omega}(M^{1/6})$ and $\mathsf{D}^{p^{3/2}\|} = O(\mathsf{Q}^{p\|}(f)^3)$ for any p.
- Thus, the conjecture must be proven/falsified in the regime where bs and p are subpolynomial but superconstant. Moreover, what is the smallest power of p required to show this for all total f? Analogous questions remain open for $\mathsf{D}^{p\|}$ vs $\mathsf{R}^{p\|}$ as well.

Parallel composition theorem. It is known that $\mathsf{Q}(f \circ g) = \Theta(\mathsf{Q}(f) \cdot \mathsf{Q}(g))$ for boolean decision functions, a widely applicable and powerful result [Rei11b]. Is there an analogous theorem for p-parallel query complexity? In particular, do we have

$$\mathsf{Q}^{p\|}(f \circ g) = \Theta\left(\min_{q \in [p]} \mathsf{Q}^{q\|}(f) \cdot \mathsf{Q}^{\lceil p/q \rceil\|}(g)\right)$$

for boolean f, g? Such a result would suffice to reproduce our lower bound for $\textsc{And} \circ \textsc{Or}$, and likely be widely applicable for understanding parallel quantum query complexity. A straightforward attempt to show composition of the p-parallel adversary quantity using the reduction in [GR04] fails. This is because the partial function f' whose sequential query complexity characterizes the p-parallel complexity of f is no longer boolean.

Natural functions giving unbounded separation in rounds. Our results in Section 1.1.1 and Section 1.1.2 answer affirmatively whether it is possible for total functions to have unbounded separations in rounds, and whether this is still possible when no sequential separation exists. However, finding more natural functions to answer both these questions remains open and could help in understanding the interplay between quantum algorithms and parallel queries better.

1.3 Related work

Prior work on parallel quantum algorithms in the query model has primarily demonstrated the need for quantum depth in solving certain problems.

Cryptography There has been a recent line of work studying problems which can only be efficiently solved by quantum algorithms with high depth [CCL23, ACC+23, CH22, CH23, CM20], constructing cryptographic proofs of quantum depth. In this vein, Chung et al [CFHL21] and Blocki et al [BLZ21] show parallel quantum lower bounds against producing the output of an iterated hash function to give a cryptographic proof of sequential work that is secure against quantum computers.

Partial functions Burchard [Bur19] gives a characterization of parallel quantum approximate counting, showing that, approximating to multiplicative error ϵ, the number K of marked elements in a list of size N requires $\Omega\left(\frac{\sqrt{N}}{\epsilon\sqrt{pK}}\right)$ p-parallel quantum queries. Girish et al [GSTW23] constructs a partial function which exhibits a large total query separation between r-adaptive and $r-1$-adaptive quantum algorithms for any constant r.

Circuit complexity Another setting one could study the intersection of quantum and parallelism is in the circuit model. For instance, Cleve and Watrous [CW00] show how to implement the quantum fourier transform on n qubits in $O(\log n)$ depth, allowing for a highly parallel implementation of Shor's factoring algorithm. There are known examples of relational problems which can be computed in constant quantum depth, yet require at least logarithmic depth for classical circuits [BGK18, WKST19, WP23]. More generally, quantum circuit classes such as QAC_n and QNC_n have been extensively studied [GHMP02, HŠ05, TT16].

1.4 Organization

Our paper consists of 4 main results, one corresponding to each point in the abstract. For each of our results:

- The theorem statement is given in the Main Results subsection (Section 1.1),
- An intuitive explanation of the technical aspects of each result is given in the Technical Overview section (Section 2)
- The required definitions are given, the main theorem is restated, more detailed explanations and the lemmas and proofs for each result are given in their respective sections (Sections 4 to 7).

We provide links to each of the above for convenience:

Result Name	Result Statement	Technical Overview	Full Details
Unbounded parallel separations for total functions	Section 1.1.1	Section 2.1	Section 4
Unbounded genuine parallel separations	Section 1.1.2	Section 2.2	Section 5
Separations with two layers of adaptivity	Section 1.1.3	Section 2.3	Section 6
Parallel quantum lower bound framework and applications	Section 1.1.4	Section 2.4	Section 7

Preliminaries are provided in Section 3 and auxiliary proofs are provided in Appendices A to C

2 Technical overview

In this section, we outline the techniques used in showing our main results. For our results involving separations between query complexity measures, we will primarily describe the techniques involved in separating quantum from randomized query complexity measures unless the techniques for separating randomized and deterministic query complexity measures are significantly different.

2.1 Unbounded parallel separations for total functions

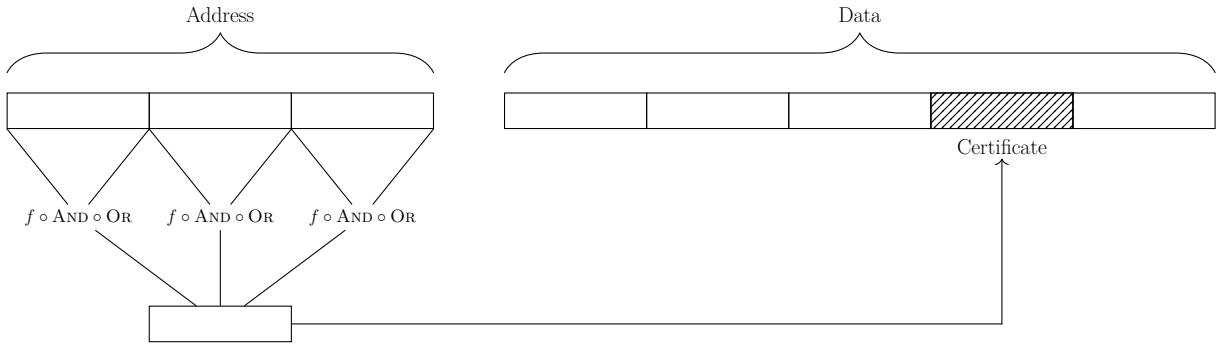

Figure 2: The canonical cheat sheet function f_{ccs} which lifts a partial function f to a total function, while retaining some of the speedup of f.

The cheatsheet framework [ABK16] lifts a partial function f, potentially with an exponential quantum query advantage to a total function that retains some of the advantage (although now polynomial). The natural way to do this is to define a total function such that when the input is in the domain of f, answer as f does, and when it is not answer 0. The cheat sheet framework does this "domain check" in a clever way so that the quantum advantage is not lost in the process. As shown in Figure 2, the input for the total function f_{ccs} is divided into two components, the Address section and the Data section. The partial function f is composed with a total function that has low certificate complexity (defined in Section 3), which in this case is $\text{And} \circ \text{Or}$. This composition $f \circ \text{And} \circ \text{Or}$ produces a single bit. This is repeated such that enough bits to address a block in the data component are obtained. If this block contains the certificate (or "cheatsheet") for the $\text{And} \circ \text{Or}$'s, and these certify that the input of f is in its domain, then the output of f_{ccs} is 1, else, the output is 0. As the data component is large, an algorithm is forced to solve the partial function f to obtain the address of the block, and can use the certificate present there to check whether the input of f is in the domain (and thus does not need to read the full Address to perform the domain check).

Thus, computing a cheatsheet function involves

(1) Solving $f \circ \text{And} \circ \text{Or}$ to get the certificate location,

(2) Reading out the certificate,

(3) Checking the validity of the input of f.

For concreteness, consider the canonical cheatsheet function defined by Aaronson et al [ABK16] composing $f = \text{Forrelation}$ [Aar10, AA15] on N bits with $\text{And} \circ \text{Or}$ on N^2 bits. This function was originally constructed to exhibit a superquadratic separation between sequential quantum and

randomized query complexities. Observe that with $p = N^2$ parallelism, solving any given instance of the internal $\textsc{And} \circ \textsc{Or}$ can be done in a single p-parallel query by reading out all the inputs. A quantum algorithm can utilize this to quickly solve the whole composition, as Forrelation is quantumly easy. However, any classical algorithm would seem to need $\mathsf{R}(\textsc{Forrelation})$ p-parallel queries to solve this composition since it requires at least $p \cdot \mathsf{R}(\textsc{Forrelation})$ sequential queries. Thus, with p parallelism, step (1) is quantum efficient but not classically efficient. We show that for the canonical choice of parameters, steps (2) and (3) are also possible with few p-parallel queries, resulting in the desired separation.

The same technique can be used to give an exponential total function separation between randomized and deterministic parallel query complexity.

Our result, combined with an aforementioned result of [JMW17], which states that $\mathsf{D}^{p\|}(h) = \mathsf{poly}(\mathsf{Q}^{p\|}(h))$ for all total h and $p = O(\mathsf{bs}(h)^{1-\epsilon})$ with $\epsilon > 0$, provides a new way of upper bounding the block sensitivity of total Boolean functions. In particular, using our separation and a lower bound argument for the canonical cheatsheet function, we characterize its block sensitivity.

2.2 Unbounded genuine parallel separations

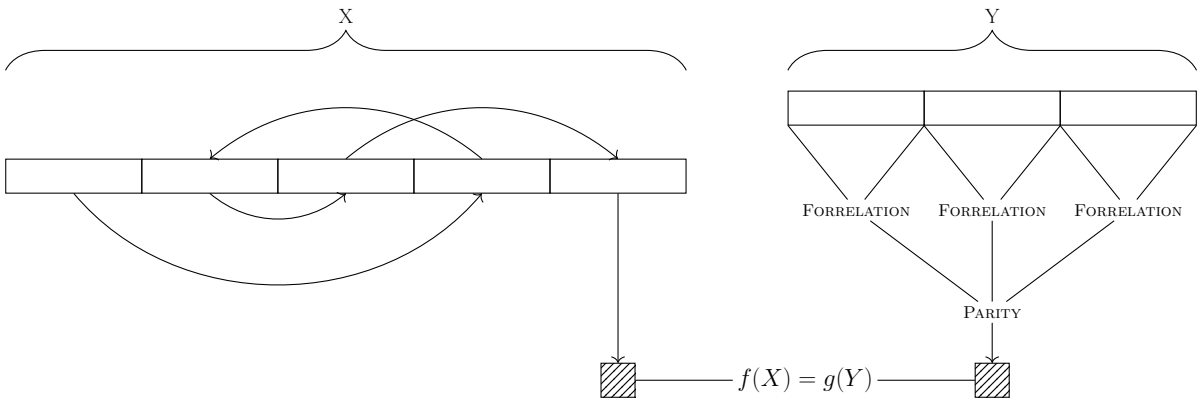

Figure 3: Partial function that admits unbounded genuine parallel separation

Is some polynomial sequential separation necessary for unbounded parallel quantum speedup, or can the advantage emerge purely from a quantum algorithm's ability to utilize parallelism (which we call a genuine parallel advantage)? We argue that the latter can indeed be the case, by first describing our construction for the partial function h that allows proving Theorem 3. As shown in Figure 3, the input to h here involves two components X and Y. The function f requires a sequence of adaptive queries to solve (in either model) whereas the function g achieves quantum-randomized separation and can be fully parallelized. We are promised that the output of both will be the same, so any algorithm can choose to solve either. We then argue that any algorithm will require solving at least one of f and g.

For concreteness, let $\epsilon > 0$ and suppose that $\mathsf{Q}(f) = \Theta(\mathsf{R}(f)) = \Theta(N^\epsilon)$ and $\mathsf{R}^{p\|}(f) = \Omega(N^\epsilon)$, and $\mathsf{Q}^{p\|}(g) = \Theta(N^\epsilon)/p$ and $\mathsf{R}^{p\|}(g) = \Theta(N^{2\epsilon})/p$ for small enough p[§]. Then, a randomized sequential algorithm can choose to solve f, while a quantum sequential algorithm will not benefit from choosing to solve either f or g. Thus, $\mathsf{R}(h) = O(N^\epsilon)$ and we would have $\mathsf{Q}(h) = \Omega(N^\epsilon)$. Similarly, for $p = \mathsf{Q}(g)$, a p-parallel quantum algorithm can choose to solve g, while a p-parallel randomized

[§]For more concreteness, one may think of f as the pointer chasing function (see Problem 15) with chain length N^ϵ and g as the composition of parity on N^ϵ bits composed with $\textsc{Forrelation}$ on $N^{2\epsilon}$ bits as shown in Figure 3.

algorithm will not benefit from choosing to solve either f or g. Therefore, $\mathsf{Q}^{p\|}(h) = 1$ and we would have $\mathsf{R}^{p\|} = \Omega(N^{\epsilon})$.

To show our desired randomized parallel and quantum lower bounds, we consider the general problem, which we call COR, where we are given inputs X and Y to arbitrary functions f and g respectively along with the promise that $f(X) = g(Y)$, our goal is to output $f(X) = g(Y)$ (see Problem 35). We use the hybrid argument to establish that any randomized parallel algorithm that solves $\textsc{cor}(f, g)$ will be able to either distinguish an input sampled from a hard 0-distribution for f from an input sampled from a hard 1-distribution for f, or will succeed at a similar distinguishing task for g. Therefore, we must have $\mathsf{R}^{p\|}(\textsc{cor}(f, g)) = \Omega(\min(\mathsf{R}^{p\|}(f), \mathsf{R}^{p\|}(g)))$. For the quantum lower bound, we show that that for any adversary matrices $\Gamma^{(f)}$ and $\Gamma^{(g)}$ for f and g respectively, the matrix $\Gamma = \Gamma^{(f)} \otimes \Gamma^{(g)}$ is an adversary matrix for $\textsc{cor}(f, g)$ and $\max_i \|\Gamma_i\| = \max\left(\max_i \left\|\Gamma_i^{(f)}\right\|, \max_i \left\|\Gamma_i^{(g)}\right\|\right)$[¶] (see Section 3.4 and Lemma 39). It follows that $\mathsf{Adv}(\textsc{cor}(f, g)) = \Omega(\min(\mathsf{Adv}(f), \mathsf{Adv}(g)))$, which implies the desired lower bound (see eq. (1) and Theorem 17).

Next, we describe a way to totalize the partial function h that we constructed in the previous section while maintaining some of its properties. We will use the cheatsheet framework. However, we will need a function with relatively small certificate complexity that does not admit any significant quantum speed-up so $\textsc{And} \circ \textsc{Or}$ will not work. Fortunately, [ABK16] found a function, which they called Bkk (see Problem 11), that satisfies $\mathsf{Q}(\textsc{Bkk}_N) = \tilde{\Theta}(\textsc{Bkk}_N) = \tilde{\Theta}(N)$ and $\mathsf{C}(\textsc{Bkk}_N) = \tilde{O}(\sqrt{N})$. We compose h on N bits with Bkk on N^2 bits, and then plug it into the cheatsheet framework. The desired upper bounds for $\mathsf{Q}^{p\|}$ and R in Theorem 4 are relatively straightforward and follows from the discussion in the previous sections. The quantum lower bound follows from quantum query complexity composition theorem and results in [ABK16]. For the randomized parallel lower bound, we show a composition theorem where the inner function is Bkk (see Theorem 69).

The same lower bound techniques do not follow for the analogous separation between the randomized and deterministic query complexity measures (see Theorem 43). In particular, there is no general randomized composition theorem. Fortunately, for our constructions, the known randomized composition theorems suffice (see Theorem 19). For the deterministic parallel lower bound, we show a general composition theorem when the degree of the inner function is almost full, which might be of independent interest.

[¶]All the max are over relevant indices.

2.3 Separations with two layers of adaptivity

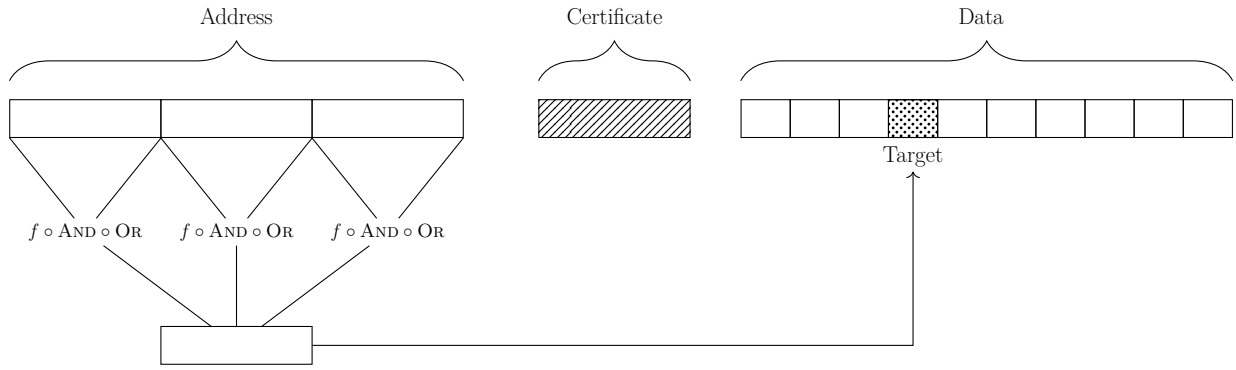

Figure 4: A framework that converts a partial function separation with one layer of adaptivity to a total function with two layers of adaptivity.

As we noted earlier (see Theorem 2), the cheat sheet framework can be employed to show a polynomial separation between $\mathsf{Q}^{3\perp}$ and $\mathsf{R}^{3\perp}$. We modify this framework to show a polynomial separation between $\mathsf{Q}^{2\perp}$ and $\mathsf{R}^{2\perp}$. As shown in Figure 4, our strategy is essentially to separate the cheatsheet into two parts: the first part contains the certificate, and the second just contains a long data string with one relevant index. Let f be a partial function with a polynomial separation between $\mathsf{Q}^{1\perp}(f)$ and $\mathsf{R}^{1\perp}(f)$. Informally, embedding f in our framework results in a function that requires (1) verifying the input to the f is in the domain (using the certificate) and if so, (2) outputting the bit of the data string present at the address obtained by solving the f.

For the formal problem description, see Problem 51.

The quantum algorithm easily succeeds as follows. In the first round, it can read the certificate as well as solve f to obtain the address. In the second round, it verifies the certificate and queries the bit at the right address in the data string, hence is able to output the result. The randomized algorithm is unable to succeed in the same amount of parallelism because after the first round, it cannot compute f, so it would not find the right address by the first round, and can only make a query to the right address by the second round if it guesses the location correctly, which happens with very low probability. We also note that there is a caveat: the structure of a zero-certificate might be easy to distinguish from the structure of a one-certificate, allowing the algorithm to infer the inputs to the partial function from just the structure of the certificates. In that case, the randomized algorithm can use this to compute the partial function in the first round itself. To prevent this from happening, we use "bi-certificates" instead of certificates, where a "bi-certificate" corresponds to a set of indices that could certify both a zero and a one instance. This prevents the randomized algorithm from learning any information just by knowing the certificate.

Therefore, Problem 51 presents a framework to lift a partial function $\mathsf{Q}^{1\perp}$ vs $\mathsf{R}^{1\perp}$ separation to a total function $\mathsf{Q}^{2\perp}$ vs $\mathsf{R}^{2\perp}$ separation (see Theorem 55). The analogous result holds for randomized vs deterministic algorithms (see Theorem 52).

2.4 Quantum parallel lower bound framework and applications

There are few known techniques for lower bounding parallel quantum query complexity. Consider for instance lower bounding the quantum parallel query complexity for the balanced $\textsc{And} \circ \textsc{Or}$ function,

which is an example of a read-once formula. It is well known that this function has sequential quantum query complexity $\Theta(\sqrt{N})$ [BS04, Rei11a]. Since both $\mathsf{Q}^{p\|}(\textsc{And})$ and $\mathsf{Q}^{p\|}(\textsc{Or})$ are $\Omega\left(\sqrt{N/p}\right)$, a natural guess for $\mathsf{Q}^{p\|}(\textsc{And} \circ \textsc{Or})$ is $\Omega\left(\sqrt{N/p}\right)$.

The bound $\mathsf{Q}(\textsc{And}\circ\textsc{Or}) = \Omega(\sqrt{N})$ was shown using the (sequential) combinatorial adversary method. However, as mentioned in Section 1.1.4, the corresponding parallel version fails to show a lower bound better than $\Omega\left(\sqrt{\left\lceil \frac{\mathsf{C}_0(f)}{p} \right\rceil \left\lceil \frac{\mathsf{C}_1(f)}{p} \right\rceil}\right)$ for any function f (see Theorem 58). This means that the best lower bound this method could help prove for $\textsc{And} \circ \textsc{Or}$ is $\Omega(\sqrt{N}/p)$, since $\mathsf{C}_0(\textsc{And} \circ \textsc{Or}) = \mathsf{C}_1(\textsc{And} \circ \textsc{Or}) = \sqrt{N}$.

We use the parallel spectral adversary method of [JMW17] (see Theorem 17), which is known to be optimal, to derive a method that is easier to apply and is sometimes stronger than the combinatorial adversary method (see Theorem 6). For the case of read-once formulas, we use the adversary sets of [BS04] to construct an adversary matrix Γ. It is easy to lower bound $\|\Gamma\|$ by a simple counting argument, but upper bounding $\|\Gamma_S\|$ for all $S \subseteq [N]$ with $|S| = p$ can be challenging. We show that if Γ satisfies the property that $\Gamma[x, y] = 0$ for all x, y that differs in more than 1 bit, then all the induced Γ_S can be rearranged to form block-diagonal matrices with blocks of size $2^p \times 2^p$. Moreover, each of these blocks are adversary matrices for some restricted function $g \in \mathcal{F}^{(f)}_{p-\text{res}}$ (Figure 5), where a restricted version of f is one where all but p input bits are fixed and known by the algorithm. Thus, we know that the spectral norm of these blocks is upper bounded by $\mathsf{Q}(g)$ (up to the normalizing factor of $\max_{i\in[2^p]} \|\Gamma_i\|$, and with a max taken over all $g \in \mathcal{F}^{(f)}_{p-\text{res}}$), which we can use to upper bound the spectral norm of any Γ_S. In our case, noting that g is a read-once formula of size p, we have $Q(g) = O(\sqrt{p})$ and we obtain the desired lower bound of $\Omega\left(\sqrt{N/p}\right)$. Hence, we are able to reduce the task of finding parallel lower bounds to the potentially easier task of finding sequential upper bounds.

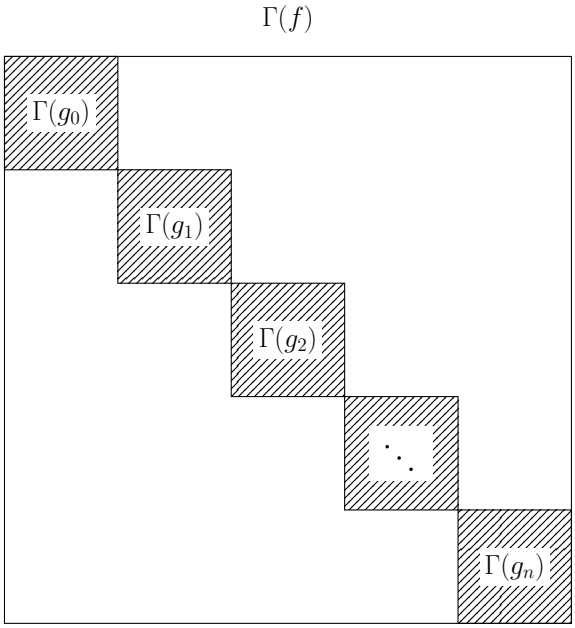

Figure 5: Nearest neighbor adversaries can be block-diagonalized, where the blocks are adversary matrices for restricted versions of the function f.

3 Preliminaries

In this section, we will review some of the notions related to quantum query complexity, adversarial methods and the cheat sheet framework of [ABK16].

3.1 Notation

For integers $N_1 < N_2$, we use $[N_1, N_2]$ to denote the set $\{N_1, N_1 + 1, \ldots, N_2\}$. We also use $[N]$ for $[1, N]$ for simplicity. For a set $S \subseteq [N]$, we will use $S^{\complement}$ to denote the set $[N] \setminus S$. For strings x, y, we use $x \parallel y$ to denote concatenation of x with y, $x[i]$ to denote the character at the ith position in x and $x[i :]$ to denote the substring of x starting from the ith position. For functions f, g, we use $f \circ g$ to denote the composition of f with g, $\tilde{O}(f) = O(f \cdot \mathsf{polylog}(f))$, $\tilde{\Omega}(f) = \Omega(f/\mathsf{polylog}(f))$ and $\tilde{\Theta}(f) = g$ iff $g = \tilde{O}(f)$ and $g = \tilde{\Omega}(f)$. Note that our use of $\tilde{O}$ and $\tilde{\Omega}$ are such that logarithmic factors in all asymptotic variables are suppressed, unless otherwise stated. We use $\mathbb{I}[\text{prop}]$ to denote the indicator variable which is 1 if prop is true, and 0 otherwise.

3.2 Boolean complexity measures and query complexity

Let $f : \mathcal{D} \to \{0, 1\}$ be a Boolean function with domain $\mathcal{D} \subset \{0, 1\}^N$. We say that f is total if $\mathcal{D} = \{0, 1\}^N$; otherwise, we say it is partial. Sometimes, instead of considering a Boolean alphabet $\{0, 1\}$, we will consider larger alphabets of size up to $N^{\mathsf{poly}\log N}$. However, such functions always correspond to functions with Boolean alphabet, up to logarithmic overhead factors, in the relevant complexity measures by a straightforward reduction.

We recall the combinatorial notions of certificate complexity and block sensitivity for total Boolean functions. A certificate for f on input $x \in \{0, 1\}^n$ is a set $S \subseteq [n]$ such that $f(y) = f(x)$ for all $y \in \{0, 1\}^n$ with $y_S = x_S$. Let $\mathsf{C}_x(f)$ denote the size of the smallest certificate for f on input x. The certificate complexity of f, denoted $\mathsf{C}(f)$, is $\max_{x \in \{0,1\}^n} \mathsf{C}_x(f)$.

For any $x \in \{0,1\}^n$ and $S \subseteq [n]$, let $x^{(S)}$ be the string x with the characters at positions in S flipped. For a given input $x \in \{0,1\}^n$ of f, call a subset of indices S a sensitive block if $f(x) \neq f(x^{(S)})$. Let $\mathsf{bs}_x(f)$ be the maximum number of disjoint sensitive blocks. The block sensitivity of f, denoted $\mathsf{bs}(f)$, is $\max_{x\in\{0,1\}^n} \mathsf{bs}_x(f)$.

A p-parallel classical query to x maps p bit strings $i_1, i_2, \ldots, i_p \in \{0,1\}^p$ to $x_{i_1}, x_{i_2}, \ldots, x_{i_p}$. Similarly, a p-parallel quantum oracle $\mathcal{O}_x^{p\|}$ is defined as

$$\mathcal{O}_x^{p\|} |i_1, i_2, \ldots, i_p\rangle |b_1, b_2, \ldots, b_p\rangle \to |i_1, i_2, \ldots, i_p\rangle |b_1 \oplus x_{i_1}, b_2 \oplus x_{i_2}, \ldots, b_p \oplus x_{i_p}\rangle$$

We usually call a 1-parallel (quantum) query as a sequential (quantum) query.

A p-parallel quantum algorithm is a sequence of unitary operations U_i interleaved with p-parallel queries to an input oracle $\mathcal{O}_x^{p\|}$ with an unbounded number of ancilla qubits. Any such algorithm $\mathcal{A}$ computes f if upon measuring the first qubit of the final state of $\mathcal{A}$, one receives $f(x)$ with probability at least $2/3$ for all $x \in \{0,1\}^N$. The notions of p-parallel deterministic and randomized algorithms are defined similarly.

The p-parallel deterministic (respectively randomized, quantum) query complexity of a function f, denoted $\mathsf{D}^{p\|}(f)$ (respectively $\mathsf{R}^{p\|}(f)$, $\mathsf{Q}^{p\|}(f)$), is the minimum number of classical (respectively classical, quantum) queries a p-parallel algorithm makes to compute f. We will usually refer to $\mathsf{D}^{1\|}(f)$ (respectively $\mathsf{R}^{1\|}(f)$, $\mathsf{Q}^{1\|}(f)$) as the sequential query complexity of f and denote by $\mathsf{D}(f)$ (respectively $\mathsf{R}(f)$, $\mathsf{Q}(f)$).

An orthogonal notion of complexity when queries are made in parallel is what we call k-adaptive query complexity. The k-adaptive deterministic (respectively randomized, quantum) query complexity for a function f, denoted $\mathsf{D}^{k\perp}(f)$ (respectively $\mathsf{R}^{k\perp}(f)$, $\mathsf{Q}^{k\perp}(f)$) is the minimum p such that there is an algorithm $\mathcal{A}$ that computes f using k many p-parallel deterministic (respectively randomized, quantum) queries. We will usually refer to $\mathsf{D}^{1\perp}(f)$ (respectively $\mathsf{R}^{1\perp}(f)$, $\mathsf{Q}^{1\perp}(f)$) as non-adpative query complexity.

We use $\mathsf{bs}(f)$ to denote the block sensitivity of a function and $bs^f(X)$ for the block sensitivity of an input X for the function f (see [Bd02] for formal definition). Then, bs_0 and bs_1 refer to $\max\{\mathsf{bs}^f(X) \mid f(X) = 0\}$ and $\max\{\mathsf{bs}^f(X) \mid f(X) = 1\}$ respectively.

3.3 Commonly used functions

In this section, we recall some functions that we will use in our results.

We begin with the AND $\circ$ OR function, which is a commonly used function due to its balanced 0-certificate and 1-certificate sizes (i.e. $\sqrt{N}$).

Problem 8 (AND $\circ$ OR)**.**

Input: *Oracles for N strings $X^i \in \{0,1\}^N$ for $i \in [N]$.*
Question: *Decide if there exist an i such that $X^i = 0^N$.*

We now define k-SUM. We will define its specific Boolean version, which will be comparable to the definition of BLOCK k-SUM (see Problem 10) from [ABK16]. For the quantum lower bound of [BS13, ABK16] to go through, we would need the alphabet size to be $\Omega(N^k)$. Therefore, in the definitions of k-SUM and BLOCK k-SUM, we will choose the size of each block representing an integer to be $10k \log N$.

Problem 9 (k-SUM)**.** *Let $M = N^{10k} = \Omega(N^k)$ be the alphabet size.*

Input: *An oracle for a string $X \in \{0,1\}^N$.*
Premise: *Divide X into blocks of size $10k \log N$, where each block represent a number in $[M]$.*
Question: *Decide if there are k blocks whose corresponding numbers sum to $0 \bmod M$.*

Next, we define BLOCK k-SUM. Unlike k-SUM where every block represents a number in the alphabet, only the balanced blocks represent numbers in BLOCK k-SUM, while an additional check is necessary on the unbalanced blocks to compute the function.

Problem 10 (BLOCK k-SUM [ABK16])**.** *We will have $k = \log N$ unless otherwise mentioned. Let $M = N^{10k} = \Omega(N^k)$ be the alphabet size.*

Input: *An oracle for a string $X \in \{0,1\}^N$.*
Premise: *Divide X into blocks of size $10k \log N$. A block is balanced if it has an equal number of 0s and 1s. Each balanced block represents a number in $[M]$.*
Question: *Decide if there are k balanced blocks whose corresponding numbers sum to $0 \bmod M$, and all other blocks have at least as many 1s as 0s.*

[ABK16] constructed BKK to show a maximal (quadratic) separation between quantum query complexity and certificate complexity. We state these results in a theorem below for easy referencing.

Problem 11 ((BKK) [ABK16])**.**

Input: *An oracle for a string $X \in \{0,1\}^{N^2}$.*
Question: *Compute BLOCK k-SUM$_N \circ k$-SUM$_N(X)$.*

Theorem 12 ([ABK16],Section 4)**.** *$Q(BKK_{N^2}) = \tilde{\Omega}(N^2)$ and $C(BKK_{N^2}) = \tilde{O}(N)$.*

Deutsch and Jozsa [DJ92] found a function $f : \{0,1\}^N \to \{0,1\}$ with $\mathsf{Q}(f) = O(1), \mathsf{R}(f) = O(1)$ and $\mathsf{D}(f) = \Theta(N)$. We modify this function slightly to have $\mathsf{Q}(f) = \mathsf{R}(f) = 1$ but still have $\mathsf{D}(f) = \Theta(N)$.

Problem 13 (DEUTSCH JOZSA MODIFIED)**.**

Input: *An oracle for a string $X \in \{0,1\}^N$.*
Promise: *Either $X_i = 0$ for all $i \in \{0,1\}^n$ or $|X| = N/2$ (i.e. X is balanced between 1 and 0)*
Question: *Decide which is the case.*

The 1-query randomized algorithm queries a random i, outputs 0 with probability $2/3$ if $X_i = 0$ and outputs 1 otherwise. We will sometimes refer to the DEUTSCH JOZSA MODIFIED function as dj.

The Forrelation problem, introduced by Aaronson [Aar10], can be solved with bounded error using 1 quantum query yet requires $\tilde{\Omega}(\sqrt{N})$ classical randomized queries to be solved with bounded error [AA15].

Problem 14 (FORRELATION)**.** *Let $N = 2^n$ and $X, Y : \{0,1\}^n \to \{-1,1\}$ be Boolean functions. Let*

$$\Phi_{X,Y} = \frac{1}{2^{3N/2}} \sum_{i,j \in \{0,1\}^n} X(i)(-1)^{i \cdot j} Y(j)$$

denote the "Forrelation" (fourier correlation) of X and Y.

Input: *An oracle for functions $X, Y : \{0,1\}^n \to \{-1,1\}$*
Promise: *Either $|\Phi_{X,Y}| \leq \frac{1}{100}$ or $\Phi_{X,Y} \geq \frac{3}{5}$.*
Question: *Decide which is the case.*

We will sometimes refer to the FORRELATION function as for.

The pointer chasing problem is a canonical example of a problem which does not benefit from parallel queries. It has been studied in classical communication complexity and cryptography [PRS01, CP18], and in similar contexts quantumly [KNTSZ01, BLZ21, CFHL21]. The input consists of $N = 2^n$ blocks of size n each. Each block contains a pointer to a block. $X(i)$ refers to the block pointed to by block i, and $X^k(i)$ refers to the block arrived at after following the pointer k times. Formally, the pointer chasing function is:

Problem 15 (POINTER CHASING)**.** *Let $N = 2^n$ be the instance size and k be the chain length.*

Input: *An oracle for $X : \{0,1\}^n \to \{0,1\}^n$ (represented by a string of length nN)*
Question: *Decide the last bit of $X^k(0^n)$.*

We will need the following results about the POINTER CHASING function, which we prove in Appendix C for completeness. We will sometimes use the notation $\text{POINTER}_{N,k}$ to refer to the POINTER CHASING function with instance size N and chain length k.

Theorem 16. *Let f be $\text{POINTER}_{N,k}$ for some integers N, k and let $p \in [N]$. Then,*

(i) $\mathsf{D}^{p\|}(f) = \tilde{O}(\min(k, N/p))$ *(so* $\mathsf{R}^{p\|}(f) = \tilde{O}(\min(k, N/p))$ *and* $\mathsf{Q}^{p\|}(f) = \tilde{O}(\min(k, N/p))$*),*

(ii) $\mathsf{D}^{p\|}(f) = \Omega(\min(k, N/p))$, *(iii)* $\mathsf{R}^{p\|}(f) = \Omega(\min(k, N/pk))$, *(iv)* $\mathsf{Q}(f) = \Omega(k)$.

3.4 Adversary method for parallel algorithms

The (negative weighted) adversary quantity [HLS07] is known to characterize (bounded-error) quantum sequential query complexity [Rei11b]. This result was generalized to p-parallel quantum queries by [JMW17]. For any function $f : \mathcal{D} \to \mathcal{M}$, a matrix $\Gamma \in \mathbb{R}^{|\mathcal{D}|\times|\mathcal{D}|}$ is an adversary matrix for f if Γ is symmetric and for any x, y if $f(x) = f(y)$, then $(\Gamma)_{xy} = 0$. Alternatively, we say that any $\Gamma \in \mathbb{R}^{X\times Y}$ is an adversary matrix for f where $X = f^{-1}(0)$ and $Y = f^{-1}(1)$. For any adversary matrix Γ and any set $S \subset [N]$, let Γ_S be defined as

$$(\Gamma_S)_{xy} \coloneqq \begin{cases} (\Gamma)_{xy} & \text{if } x_S \neq y_S \\ 0 & \text{otherwise} \end{cases}$$

for all row indices x and column indices y of Γ.

The p-parallel adversary quantity for f is defined as

$$\mathsf{Adv}^{p\|}(f) \coloneqq \max_{\Gamma} \frac{\|\Gamma\|}{\max_{S\subset[N],|S|=p} \|\Gamma_S\|} \tag{1}$$

where the maximization is over all adversary matrices for f.

We state the correspondence between the p-parallel quantum query complexity and the p-parallel adversary quantity for later reference.

Theorem 17 (Parallel quantum query complexity characterization [JMW17])**.** *For any function* $f : \{0,1\}^N \to \mathcal{M}$, $\mathsf{Q}^{p\|}(f) = \Theta(\mathsf{Adv}^{p\|}(f))$.

The positive weighted adversary method was originally formulated by [Amb02] combinatorially, which was later generalized to allow for negative weights [HLS07]. The latter version characterizes bounded error quantum query complexity up to constant factors [Rei11b], and both can be generalized to p-parallel quantum queries (where once again the negative weighted version is tight up to constants). We note that the combinatorial version is not generally tight (and is known to be not tight in the sequential case [Zha04]), but can be easier to work with.

Theorem 18 (Parallel combinatorial adversary method [GR04, Bur19])**.** *Let* $f : \{0,1\}^N \to \mathcal{M}$ *be any function and* S *be a subset of* $[N]$ *such that* $|S| \leq p$*. For any* X *and* Y *be such that* $f(x) \neq f(y)$ *for all* $x \in X, y \in Y$ *and any relation* $R \subseteq X \times Y$*, define*

$$w(x,y) := \begin{cases} 1 & \text{if } (x,y) \in R \\ 0 & \text{otherwise} \end{cases}$$

$$w_x := \sum_y w(x,y) \qquad w_{x,S} := \sum_{x,S \mid x_S \neq y_S} w(x,y) \qquad w_y := \sum_x w(x,y) \qquad w_{y,S} := \sum_{y,S \mid x_S \neq y_S} w(x,y)$$

$$m := \min_{x \in X} w_x \qquad l := \max_{x,S} w_{x,S} \qquad m' := \min_{y \in Y} w_y \qquad l' := \max_{y,S} w_{y,S}$$

Then, $\mathsf{Q}^{p\|}(f) = \Omega\left(\max_{X,Y,R} \sqrt{\frac{mm'}{\ell\ell'}}\right)$.

Sequential deterministic and quantum query complexities of composed Boolean functions is easy to express in terms of the complexity of the functions being composed due to the work of [Rei11b, Kim12, Tal13, Mon13]. No such general result is known for randomized query complexity. In fact, when the relevant functions are partial, it is known to be not true [BDB20]. However, when the outer function is And or Or (or a composition of the two), such a result is known [ABK16]. We summarize these results in the following theorem.

Theorem 19 (Query complexity composition theorem)**.** *Let* $f : \mathcal{D}_1 \to \{0,1\}$ *and* $g : \mathcal{D}_2 \to \{0,1\}$ *be any Boolean functions with* $\mathcal{D}_1 \subseteq \{0,1\}^{N_1}$ *and* $\mathcal{D}_2 \subseteq \{0,1\}^{N_2}$ *for some integers* N_1, N_2*. Then,*

1. $\mathsf{D}(f \circ g) = \Theta\left(\mathsf{D}(f) \cdot \mathsf{D}(g)\right)$ [Tal13, Mon13],
2. $\mathsf{R}(f \circ g) = \tilde{O}\left(\mathsf{R}(f) \cdot \mathsf{R}(g)\right)$*, and if* $g \in \{\mathsf{And}, \mathsf{Or}\}$ *or* $\mathsf{R}(f) = \Theta(N_1)$*, then* $\mathsf{R}(f \circ g) = \Omega\left(\mathsf{R}(f) \cdot \mathsf{R}(g)\right)$ [ABK16, BDB20, CKM+23],
3. $\mathsf{Q}(f \circ g) = \Theta\left(\mathsf{Q}(f) \cdot \mathsf{Q}(g)\right)$ [Rei11b, Kim12].

3.5 Cheat sheet framework

The cheat sheet framework of Aaronson et al. [ABK16] was primarily designed to construct separations between various Boolean complexity measures for total functions [ABK16, ABBD+16], especially a super-quadratic separation between the quantum and randomized query complexities.

This framework allows for constructing a total Boolean function f_{cs} from a partial function f that preserves some of the desirable properties of f.

Definition 20 (Cheat sheet framework [ABK16])**.** *Let* $f : \mathcal{D} \to \{0,1\}$ *be any Boolean function with* $\mathcal{D} \subseteq \{0,1\}^F$ *and let* c *be an integer. We define a total function* $f^c_{\mathsf{cs}} : \{0,1\}^{cF} \times \{0,1\}^{M2^c} \to \{0,1\}$ *constructed from* f *as follows where* M *is the size of the "cheatsheet" and is the number of bits required*

to certify if c inputs to f are in the domain of f. On input $z = (x^{(1)}, x^{(2)}, \cdots, x^{(c)}, y^{(1)}, y^{(2)}, \cdots y^{(2^c)})$, f^c_{cs} where $|x^{(i)}| = F$ and $|y^{(i)}| = M$, f^c_{cs} outputs 1 iff the following conditions are satisfied (where the condition 2 is only relevant when the condition 1 is true).

1. *For each $i \in [c]$, $x^{(i)} \in \mathcal{D}$.*
2. *Let $\ell \in \{0,1\}^c$ be a binary string such that $\ell_i = f(x^{(i)})$. Then, $y^{(\ell)}$ certifies condition 1, and that $\ell_i = f(x^{(i)})$ for all $i \in [c]$.*

For any input $z = (x^{(1)}, x^{(2)}, \cdots, x^{(c)}, y^{(1)}, y^{(2)}, \cdots y^{(2^c)})$ to f^c_{cs}, we will usually refer to $x^{(1)}, x^{(2)}, \cdots, x^{(c)}$ and $y^{(1)}, y^{(2)}, \cdots y^{(2^c)}$ as the *address* and *data* parts of z respectively. Moreover, we will refer to $y^{(i)}$ as the i^{th} cheat sheet for input z. The cheat sheet framework is usually embedded with a function f that has small certificate complexity so that the complexity of checking the condition 2 in Definition 20 is bounded by the complexity of computing f. Moreover, we would usually want the value of c to be small enough that so that the complexity of computing c instances of f is not much more than the complexity of computing one instance but large enough so that any algorithm is forced to compute all c instances of f before querying any information in the cheat sheet. For these reasons, we define a canonical version of cheat sheet function that will be very handy for us.

Definition 21 (Canonical cheat sheet function)**.** *Let $f : \mathcal{D} \to \{0,1\}$ be any Boolean function with $\mathcal{D} \subseteq \{0,1\}^N$ and let $f' : \mathcal{D}' \to \{0,1\}$ be the function $f \circ \mathsf{And}_N \circ \mathsf{Or}_N$ with $\mathcal{D}' \subseteq \{0,1\}^{N^3}$. Let $c = 10 \log N$, $c' = O(\log N)$ be the space needed to store a query address-value pair for f, and V be the size of the cheatsheet. Then, define the canonical cheat sheet function $f_{\mathsf{ccs}} : \{0,1\}^{cN^3} \times \{0,1\}^{V2^c} \to \{0,1\}$ as $(f')^c_{\mathsf{cs}}$*

Notice that f_{ccs} is a total Boolean function and M is $O(cc'N^2) = O(N^2 \log^2 N)$ since the certificate complexity of $\mathsf{And}_N \circ \mathsf{Or}_N$ is only N. Intuitively, the cheat sheet framework would not help any algorithm perform more efficiently since it would need to compute c independent instances of f before it could access the relevant cheat sheet.

Theorem 22 ([ABK16])**.** *Let $f : \mathcal{D} \to \{0,1\}$ be any Boolean function with $\mathcal{D} \subseteq \{0,1\}^N$ and let $c = \mathsf{poly} \log N$ be an integer. Then, $\mathsf{D}(f^c_{\mathsf{cs}}) = \tilde{\Omega}(\mathsf{D}(f))$, $\mathsf{R}(f^c_{\mathsf{cs}}) = \tilde{\Omega}(\mathsf{R}(f))$ and $\mathsf{Q}(f^c_{\mathsf{cs}}) = \tilde{\Omega}(\mathsf{Q}(f))$.*

4 Unbounded parallel separations for total functions

In this section, we demonstrate the power of the cheat sheet framework by constructing exponential separations between parallel query algorithms for total functions. Our separations use the cheat sheet (Definition 21) framework, which takes as input a partial function f and returns a total function. Substituting different f gives us the separations for quantum vs randomized and randomized vs deterministic. First in Section 4.1 we give upper bounds for this canonical cheat sheet function. We use these upper bounds in Section 4.2 to show the unbounded parallel separation between randomized and deterministic complexities, and in Section 4.3 to show the unbounded parallel separation between quantum and randomized complexities. Finally, in Section 4.4 we prove that the block sensitivity of the canonical cheat sheet function is nearly equal to the parallelism at which unbounded speedup occurs, thus refuting Conjecture 1 in the strongest possible way.

4.1 Upper bounds in the cheatsheet framework

We start off by proving deterministic and randomized sequential upper bounds for any partial function f composed with a total function g plugged into a cheat sheet.

Proposition 23. *Let $f : \mathcal{D} \to \{0,1\}$ be any (partial) Boolean function with $\mathcal{D} \subseteq \{0,1\}^N$, let $g : \{0,1\}^{N^2} \to \{0,1\}$ be any total Boolean function with $\mathsf{C}(g) = \tilde{O}(N)$, and let $c = 10 \log N$. Let $(f \circ g)^c_{\mathsf{cs}}$ be the cheatsheet version of $f \circ g$ (see Definition 20). Then,*

(i) $\mathsf{D}((f \circ g)^c_{\mathsf{cs}}) = \tilde{O}(\max(\mathsf{D}(f \circ g), N^2))$,

(ii) $\mathsf{R}((f \circ g)^c_{\mathsf{cs}}) = \tilde{O}(\max(\mathsf{R}(f \circ g), N^2))$.

Proof. We will only prove part (i) since the proof of part (ii) is very similar. We describe a deterministic algorithm that computes $(f \circ g)^c_{\mathsf{cs}}$ using at most $\tilde{O}(\max(\mathsf{D}(f \circ g), N^2))$ deterministic queries as follows. Let $z = (x^{(1)}, x^{(2)}, \cdots, x^{(c)}, y^{(1)}, y^{(2)}, \cdots y^{(2^c)})$ be an input to $(f \circ g)^c_{\mathsf{cs}}$.

Notice that we can compute the outputs of each of the c instances of $f \circ g$ on respective inputs $x^{(1)}, x^{(2)}, \cdots, x^{(c)}$ using $\tilde{O}(\mathsf{D}(f \circ g))$ queries to $(f \circ g)^c_{\mathsf{cs}}$.

As in Definition 20, let ℓ be the length c binary string such that $\ell_i = (f \circ g)(x^{(i)})$. As $\mathsf{C}(g) = \tilde{O}(N)$, the length of each cheatsheet $y^{(i)}$ is $\tilde{O}(N^2)$ so, in particular, we can query the contents of $y^{(\ell)}$ with $\tilde{O}(N^2)$ deterministic queries.

Given $y^{(\ell)}$, we only need to query $\tilde{O}(N^2)$ indices (i.e. those pointed by $y^{(\ell)}$) of $(f \circ g)^c_{\mathsf{cs}}$ to verify that $x^{(i)}$ is in the domain of $f \circ g$. Furthermore, since $y^{(\ell)}$ certifies each input bit to each of the c instances of f, it automatically certifies that $\ell_i = (f \circ g)(x^{(i)})$ for all $i \in [c]$, and that each $x^{(i)}$ is in the domain of $f \circ g$ so both conditions 1 and 2 in Definition 20 are satisfied. The desired result follows. □

Next, we prove randomized and quantum upper bounds for any partial function f composed with a total function g plugged into a cheat sheet.

Proposition 24. *Let $f : \mathcal{D} \to \{0,1\}$ be any (partial) Boolean function with $\mathcal{D} \subseteq \{0,1\}^N$, let $g : \{0,1\}^{N^2} \to \{0,1\}$ be any total Boolean function with $\mathsf{C}(g) = \tilde{O}(N)$, and let $c = 10 \log N$. Let $(f \circ g)^c_{\mathsf{cs}}$ be the cheatsheet version of $f \circ g$ (see Definition 20) with cheatsheet size V[‖].*

(i) If $\mathsf{R}^{q\|}(f) = k$ and $p = \max(cqN^2 \log N, V) = \tilde{\Theta}(qN^2)$, then $\mathsf{R}^{p\|}((f \circ g)^c_{\mathsf{cs}}) \leq k + 2$ and $\mathsf{R}^{k+2\perp}(f \circ g)^c_{\mathsf{cs}}) \leq p$.

(ii) If $\mathsf{Q}^{q\|}(f) = k$ and $p = \max(cqN^2 \log N, V) = \tilde{\Theta}(qN^2)$, then $\mathsf{Q}^{p\|}((f \circ g)^c_{\mathsf{cs}}) \leq k + 2$ and $\mathsf{Q}^{k+2\perp}(f \circ g)^c_{\mathsf{cs}}) \leq p$.

Proof. We will only prove part (i) since the proof of part (ii) is very similar. We describe a randomized algorithm that computes $(f \circ g)^c_{\mathsf{cs}}$ using at most $k + 2$ p-parallel queries, which is sufficient to prove both $\mathsf{R}^{p\|}((f \circ g)^c_{\mathsf{cs}}) \leq k + 2$ and $\mathsf{R}^{k\perp}(f \circ g)^c_{\mathsf{cs}}) = \tilde{O}(p)$. Let $z = (x^{(1)}, x^{(2)}, \cdots, x^{(c)}, y^{(1)}, y^{(2)}, \cdots y^{(2^c)})$ be an input to $(f \circ g)^c_{\mathsf{cs}}$.

Let $p' = \lfloor p/cq \rfloor \geq N^2 \log N$. Notice that we can compute the whole input to some g in $f \circ g$ using 1 p'-parallel query to $(f \circ g)^c_{\mathsf{cs}}$. Thus, by running the k-query q-parallel randomized algorithm for f where each query to f makes 1 p'-parallel query to $(f \circ g)^c_{\mathsf{cs}}$ allows us to compute $f \circ g$ using k many qp'-parallel queries with probability $1 - \Theta(1/c)$. Therefore, as $cqp' \leq p$, we can compute c instances of $f \circ g$ on respective inputs $x^{(1)}, x^{(2)}, \cdots, x^{(c)}$ in parallel using k many p-parallel queries with probability $2/3$.

As in Definition 20, let ℓ be the length c binary string such that $\ell_i = (f \circ g)(x^{(i)})$. Since $p \geq V$ and $\mathsf{C}(g) = \tilde{O}(N)$, we can query the contents of $y^{(\ell)}$ with a single p-parallel query.

[‖]Since $\mathsf{C}(g) = \tilde{O}(N)$, the input to f has N bits and specifying each location takes $\mathsf{polylog}(N)$ bits, $V = \tilde{\Theta}(N^2)$

Given $y^{(\ell)}$, we only need to query at most $cN \cdot \mathsf{C}(g)$ indices of $(f \circ g)^c_{\mathsf{cs}}$ to verify that $x^{(i)}$ is in the domain of $f \circ g$. But this could be done using only 1 p-parallel query. Furthermore, since $y^{(\ell)}$ certifies each input bit to each of the c instances of f, it automatically certifies that $\ell_i = (f \circ g)(x^{(i)})$ for all $i \in [c]$, and also that each $x^{(i)}$ is in the domain of $f \circ g$ so both conditions 1 and 2 in Definition 20 are satisfied. It follows that $\mathsf{R}^{p\|}(g_{\mathsf{cs}}) \leq k + 2$. □

As a corollary, note that the above upper bounds hold for the canonical cheatsheet function, which we will use to show separations in the next two sections.

Corollary 25. *Let $f : \mathcal{D} \to \{0,1\}$ be any (partial) Boolean function with $\mathcal{D} \subseteq \{0,1\}^N$. Let f_{ccs} be the canonical cheatsheet function associated with f (see Definition 21) with cheatsheet size V.*

(i) If $\mathsf{R}(f) = k$ and $p = \max(cN^2 \log N, V) = \tilde{\Theta}(N^2)$, then $\mathsf{R}^{p\|}(f_{\mathsf{ccs}}) \leq k + 2$ and $\mathsf{R}^{k+2\perp}(f_{\mathsf{ccs}}) \leq p$.

(ii) If $\mathsf{Q}(f) = k$ and $p = \max(cN^2 \log N, V) = \tilde{\Theta}(N^2)$, then $\mathsf{Q}^{p\|}(f_{\mathsf{ccs}}) \leq k + 2$ and $\mathsf{Q}^{k+2\perp}(f_{\mathsf{ccs}}) \leq p$.

4.2 Separation between randomized and deterministic complexities

Using the parallel upper bound for the canonical cheatsheet function in the previous section, we show the desired unbounded separation between randomized and deterministic complexities.

Problem 26 (CHEATSHEET DEUTSCH-JOSZA)**.** *Let $f : \{0,1\}^N \to \{0,1\}$ be dj_N (see Problem 13).*

Input: *An oracle for a string $X \in \{0,1\}^M$ (with $M = \tilde{\Theta}(N^{12})$**).*
Question: *Compute $f^c_{\mathsf{ccs}}(X)$ (see Definition 21).*

Theorem 27. *Let h be the CHEATSHEET DEUTSCH-JOSZA problem. Then there exists a $p = \tilde{\Theta}(N^2)$ such that*

(i) $\mathsf{D}^{p\|}(h) = \Omega(N)$, (ii) $\mathsf{R}^{p\|}(h) \leq 3$, (iii) $\mathsf{R}^{3\perp}(h) \leq p$.

Proof. We prove each item one by one. Recall that $f = \mathsf{dj}_N$ in Problem 26 and let $g = (\textsc{And} \circ \textsc{Or})_{N^2}$.

(i) It is well-known that $\mathsf{D}(f) = \Omega(N)$ so $\mathsf{D}(f \cdot g) = \Omega(N^3)$ by Theorem 19. It follows, from Theorem 22, that $\mathsf{D}(h) = \Omega(N^3)$. Therefore, $\mathsf{D}^{p\|} = \Omega(\mathsf{D}(h)/p) = \tilde{\Omega}(N)$.

(ii) and (iii) The desired result directly follows from part (i) of Corollary 25. □

The previous theorem gives us the separation we want, but note that the input is of size $\tilde{\Theta}(N^{12})$ there. To make the actual separation more evident (and to emphasize the relatively small size of the exponent), we restate it with an input size of M in the corollary below. To make clear where this separation occurs, we also mention the block sensitivity which comes from our results in Section 4.4.

Corollary 28. *There exists a total Boolean function $h : \{0,1\}^M \to \{0,1\}$ and an integer $p \in [M]$ such that*

(i) $\mathsf{D}^{p\|}(h) = \tilde{\Omega}(M^{1/12})$, (ii) $\mathsf{R}^{p\|}(h) \leq 3$, (iii) $\mathsf{R}^{3\perp}(h) \leq p$, (iv) $\mathsf{bs}(h) = \tilde{\Omega}(p)$

Proof. It follows from Theorem 27 and Corollary 33. □

**Since a certificate of $\textsc{And} \circ \textsc{Or}$ on N^2 bits could be represented using $\tilde{\Theta}(N)$ bits, we will have the number of input bits for the function $(f \circ g)^c_{\mathsf{cs}}$ to be $\tilde{\Theta}(N^{12})$.

4.3 Separation between quantum and randomized complexities

Using the parallel upper bound for the canonical cheatsheet function in Section 4.1, we show the desired unbounded separation between quantum and randomized complexities.

Problem 29 (CHEATSHEET FORRELATION). *Let $f : \{0,1\}^N \to \{0,1\}$ be for_N (see Problem 14).*

Input: *An oracle for a string $X \in \{0,1\}^M$ (with $M = \tilde{\Theta}(N^{12})$[††]).*
Question: *Compute $f^c_{\mathsf{ccs}}(X)$ (see Definition 21).*

Theorem 30. *Let h be the CHEATSHEET FORRELATION problem. Then, there exists $p = \Theta(N^2)$ such that*

(i) $\mathsf{R}^{p\|}(h) = \tilde{\Omega}(\sqrt{N})$, *(ii)* $\mathsf{Q}^{p\|}(h) \le 3$, *(iii)* $\mathsf{Q}^{3\perp}(h) \le p$.

Proof. We prove each item one by one. Recall that $f = \mathsf{for}_N$ in Problem 26 and let $g = (\textsc{And}\circ\textsc{Or})_{N^2}$.

(i) It is well-known that $\mathsf{R}(f) = \tilde{\Omega}(\sqrt{N})$ so $\mathsf{R}(f \cdot g) = \tilde{\Omega}(N^{5/2})$ by Theorem 19. It follows, from Theorem 22, that $\mathsf{R}(h) = \tilde{\Omega}(N^{5/2})$. Therefore, $\mathsf{R}^{p\|} = \Omega(\mathsf{R}(h)/p) = \tilde{\Omega}(\sqrt{N})$.

(ii) and (iii) The desired result directly follows from part (ii) of Corollary 25. □

Analogous to the randomized vs deterministic case, the previous theorem gives us the separation we want, but note that the input is of size $\tilde{\Theta}(N^{12})$ there. To make the actual separation more evident (and to emphasize the relatively small size of the exponent), we restate it with an input size of M in the corollary below. To make clear where this separation occurs, we also mention the block sensitivity which comes from our results in Section 4.4.

Corollary 31. *There exists a total Boolean function $h : \{0,1\}^M \to \{0,1\}$ and an integer $p \in [M]$ such that*

(i) $\mathsf{R}^{p\|}(h) = \tilde{\Omega}(M^{1/24})$, *(ii)* $\mathsf{Q}^{p\|}(h) \le 3$, *(iii)* $\mathsf{Q}^{3\perp}(h) \le p$, *(iv)* $\mathsf{bs}(h) = \tilde{\Omega}(p)$

Proof. It follows from Theorem 30 and Corollary 33. □

4.4 Discussion of block sensitivity

The block sensitivity of a function $\mathsf{bs}(f)$ is intimately related with parallel exponential advantage. Jeffery et al [JMW17] show that if $p = O(bs(f)^{1-\epsilon})$ for a constant $\epsilon > 0$, then $\mathsf{Q}^{p\|}(f)$ and $\mathsf{D}^{p\|}(f)$ are polynomially related. We use Theorem 32 to give a lower bound on $\mathsf{bs}(f)$ for functions that exhibit exponential parallel advantage and show that this phenomenon occurs exactly when $p = \tilde{O}(bs(f))$. Moreover, we observe that showing an exponential parallel separation has the unexpected application of providing an upper bound for $\mathsf{bs}(f)$, and we exploit this to characterize $\mathsf{bs}(f)$ for cheat sheet functions in Corollary 34.

Below is a general theorem to lower bound the block sensitivity of cheat sheet functions.

Theorem 32. *Let $g : \mathcal{D} \to \{0,1\}$ be a partial function where $\mathcal{D} \subseteq \{0,1\}^G$ and let $h : \{0,1\}^H \to \{0,1\}$ be a total function with $\min(\mathsf{bs}_0(h), \mathsf{bs}_1(h)) = \tilde{\Omega}(B)$. Thus, $f = g \circ h : \{0,1\}^{GH} \to \{0,1\}$ is a partial function. Construct the cheat sheet function f^c_{cs} as described in Definition 20. Then, $\mathsf{bs}(f^c_{cs}) = \tilde{\Omega}(BG)$.*

[††]Since a certificate of $\textsc{And}\circ\textsc{Or}$ on N^2 bits could be represented using $\tilde{\Theta}(N)$ bits, we will have the number of input bits for the function $(f \circ g)^c_{\mathsf{cs}}$ to be $\tilde{\Theta}(N^{12})$.

Proof. We prove this by constructing an input $\mathcal{X}$ to f_{cs}^c that attains block sensitivity $\tilde{\Omega}(BG)$ as follows:

1. For each of the c copies of g, select some input $z^{(i)}$ where $|z^{(i)}| = G$ such that $z^{(i)} \in \mathcal{D}$. Denote the bits of $z^{(i)}$ as $z_j^{(i)}$.
2. For each $z_j^{(i)}$, select string $x_j^{(i)}$ such that $|x_j^{(i)}| = H$, $h(x_j^{(i)}) = z_j^{(i)}$ and $\mathsf{bs}^h(x_j^{(i)}) = \tilde{\Omega}(B)$. The concatenation of all the $x_j^{(i)}$'s constitutes the address section of our input string.
3. Let $\ell \in \{0,1\}^c$ be such that $\ell_i = f(x^{(i)})$ for all i. Then let $y^{(l)}$ contain a correct certificate. At all other $y^{(k)}$, place some spurious data.

Observe that each $x_j^{(i)}$ has $\tilde{\Omega}(B)$ disjoint sensitive blocks for the function h, and there are $\tilde{\Omega}(G)$ many $x_j^{(i)}$'s. We will argue that all of these $\tilde{\Omega}(BG)$ disjoint blocks of $\mathcal{X}$ are sensitive for f_{cs}^c. First note that by construction, $f_{cs}^c(\mathcal{X}) = 1$. Suppose one of these blocks, say one in $x_j^{(i)}$, is flipped. Because this block is sensitive for h, this causes $z_j^{(i)}$ to flip to $z_j^{(i)'}$, and we denote the new string by $z^{(i)'}$, and the new input as $\mathcal{X}'$. Then, one of three cases could occur:

1. $z^{(i)'} \notin \mathcal{D}$, which means by definition, $f_{cs}^c(\mathcal{X}') = 0$.
2. $z^{(i)'} \in \mathcal{D}$ but $g(z^{(i)'}) \neq g(z^{(i)})$. This results in a new l' but $y^{(l')}$ contains spurious data by construction, so $f_{cs}^c(\mathcal{X}') = 0$.
3. $z^{(i)'} \in \mathcal{D}$ and $g(z^{(i)'}) = g(z^{(i)})$. This time, the certificate in $y^{(l)}$ is wrong, hence $f_{cs}^c(\mathcal{X}') = 0$.

Hence, $\mathsf{bs}(f_{cs}^c) = \tilde{\Omega}(BG)$.

□

The above theorem immediately implies a lower bound for the block sensitivity of the canonical cheat sheet function.

Corollary 33. *For any partial function $f : \mathcal{D} \to \{0,1\}$ where $\mathcal{D} \subseteq \{0,1\}^N$, the canonical cheatsheet function f_{ccs} satisfies $\mathsf{bs}(f_{ccs}) = \tilde{\Omega}(N^2)$.*

Proof. We have $h = f \circ \textsc{And} \circ \textsc{Or}$, where $\textsc{And} \circ \textsc{Or}$ satisfies $\min(\mathsf{bs}_0, \mathsf{bs}_1) = \tilde{\Omega}(N)$. Thus, $g_{\mathsf{ccs}} := f_{\mathsf{ccs}}^c$ satisfies the conditions in Theorem 32 with $B = N$ and $G = N$. Hence, $\mathsf{bs}(f_{\mathsf{ccs}}) = \tilde{\Omega}(N^2)$. □

Combining the above with the upper bound we get from Jeffery et al [JMW17] and our Theorem 30, we characterize the block sensitivity for cheat sheet functions.

Corollary 34. *Let $f : \mathcal{D} \to \{0,1\}$ be a partial function such that $\mathcal{D} \subseteq \{0,1\}^N$ and $\mathsf{Q}(f) = k$ and $\mathsf{R}(f) = \tilde{\Theta}(N^\alpha)$ hold for some $0 < \alpha \leq 1$. Then, the canonical cheatsheet function g_{ccs} satisfies $\mathsf{bs}(g_{ccs}) = \tilde{\Omega}(N^2)$ and $\mathsf{bs}(g_{ccs}) = O(N^{2+\epsilon})$ for all $\epsilon > 0$.*

Proof. Fix an ϵ and let $\delta = 1 - \frac{2}{2+\epsilon}$. Using the same argument as in Theorem 30, we can show that at $p = \tilde{\Theta}(N^2)$, $\mathsf{Q}^{p\|}(f_{\mathsf{ccs}}) = O(1)$ whereas $\mathsf{R}^{p\|}(f_{\mathsf{ccs}}) = \Omega(N^\alpha)$. However, from ([JMW17], Theorem 12), we know that for all $p = O(\mathsf{bs}(f)^{1-\delta})$, $\mathsf{R}^{p\|}(h) = O\left(\mathsf{poly}\left(\mathsf{Q}^{p\|}(h)\right)\right)$. Hence, it must be the case that $p = \Omega(\mathsf{bs}(f_{\mathsf{ccs}})^{1-\delta})$ so $\mathsf{bs}(f_{\mathsf{ccs}}) = O(N^{2+\epsilon})$. Combining it with Corollary 33 proves the desired result. □

5 Unbounded genuine parallel separations

In Section 4, we demonstrated that cheatsheet functions can show an unbounded parallel speedup. However, these functions show a considerable sequential speedup too (in fact they were originally constructed in [ABK16] for this very purpose). One might wonder then if a sequential speedup is always necessary to see a parallel speedup. Our goal is to prove this is not the case, and we do this over this section and the next (Section 5). In this section, we construct a *partial* function that satisfies our properties (parallel advantage with no sequential advantage). We will use this partial function in the next section to construct a total function that accomplishes our goal. Our partial function uses the *adaptive non-adaptive*, or ANA construction, which provides a framework for accomplishing these separations.

At a high level, the ANA function constructions will consist of two components: an adaptive function f which does not parallelize, and a non-adaptive function g which parallelizes, but admits an advantage for the stronger query model. In particular, an instance of the ANA function will consist of an instance of f and an instance of g, promised that the evaluation of both will be the same. Therefore, an algorithm can choose which function it wants to solve, depending on the amount of parallelism it gets. By tuning the instance sizes, the ability of an algorithm in the stronger model to solve g allows it to parallelize, while the inability of the algorithm in the weaker model to efficiently solve g requires it to compute the ANA function only via computing f, which bottlenecks its ability to effectively use the given parallelism.

5.1 Correlated functions

Before we describe our separations, we will prove generic results relating the query complexity of a function with components promised to produce the same output with the query complexities of the components, which may be of independent interest.

Precisely, we consider the following problem of computing functions with input components correlated in their outputs.

Problem 35 ($\textsc{cor}(f,g)$)**.** *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be any (partial) Boolean functions.*

Input: *An oracle for the strings $x \in \mathcal{D}_f$ and $y \in \mathcal{D}_g$.*
Promise: *$f(x) = g(y)$.*
Question: *Output $f(x) = g(y)$.*

For an input (x, y) to $\textsc{cor}(f, g)$, we will also refer to x and y as the f-component and g-component of the input to $\textsc{cor}(f, g)$ respectively. Also, we define $X_0 = f^{-1}(0)$, $X_1 = f^{-1}(1)$, $Y_0 = g^{-1}(0)$ and $Y_1 = g^{-1}(1)$.

In the rest of this section, we aim to show that the deterministic and randomized p-parallel and quantum sequential query complexity of $\textsc{cor}(f, g)$ is the minimum of the relevant query complexities of computing f and g. We begin with the result for deterministic p-parallel query complexities.

Lemma 36. *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be any (partial) Boolean functions. Then,* $\mathsf{D}^{p\|}(\textsc{cor}(f,g)) = \min(\mathsf{D}^{p\|}(f), \mathsf{D}^{p\|}(g))$.

Proof. Let $\mathcal{A}_f$ and $\mathcal{A}_g$ be optimal deterministic p-parallel algorithms for f and g. Given an input (x, y) for $\textsc{cor}(f, g)$, running either $\mathcal{A}_f$ on x or $\mathcal{A}_g$ on y is sufficient to compute it. Thus, $\mathsf{D}^{p\|}(\textsc{cor}(f,g)) \leq 2 \cdot \min(\mathsf{D}^{p\|}(f), \mathsf{D}^{p\|}(g))$.

We now argue for the lower bound using the adversary argument. That is, for any deterministic p-parallel algorithm for $\textsc{cor}(f,g)$, we answer queries to the f-component of the input according to an adversary strategy for f and queries to the g-component of the input according to an adversary strategy for g. Now, suppose that $\mathcal{A}$ is a deterministic p-parallel algorithm for $\textsc{cor}(f,g)$ that makes $t < \min(\mathsf{D}^{p\|}(f), \mathsf{D}^{p\|}(g))$ queries. Then, it must have made less than $\mathsf{D}^{p\|}(f) \leq p$-parallel queries to the f-component of the input and less than $\mathsf{D}^{p\|}(g)$ p-parallel queries to the g-component of the input. Therefore, after t p-parallel queries, both the f-component and the g-component of the input are not determined to be 0-inputs or 1-inputs, meaning that there is a consistent 0 or 1 input to $\textsc{cor}(f,g)$. It follows that t p-parallel queries are not sufficient to compute $\textsc{cor}(f,g)$ so $\mathsf{D}^{p\|}(\textsc{cor}(f,g)) \geq \min(\mathsf{D}^{p\|}(f), \mathsf{D}^{p\|}(g))$. □

Before we consider the randomized analog of Lemma 36, we will prove the following useful proposition. For distributions $\mathcal{P}$ and $\mathcal{P}'$, we will usually say distinguishing $\mathcal{P}$ and $\mathcal{P}'$ to mean distinguishing an input x distributed according to the distribution $\mathcal{P}$ from an input x' distributed according to the distribution $\mathcal{P}'$ with probability $1/2 + \Omega(1)$.

Proposition 37. *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be any (partial) Boolean functions. Let $\mathcal{P}_0^f$, $\mathcal{P}_1^f$, $\mathcal{P}_0^g$ and $\mathcal{P}_1^g$ be any distributions over X_0, X_1, Y_0 and Y_1 respectively. Let $r_f = \mathsf{R}^{p\|}(\mathcal{P}_0^f, \mathcal{P}_1^f)$ and $r_g = \mathsf{R}^{p\|}(\mathcal{P}_0^g, \mathcal{P}_1^g)$ be the randomized p-parallel query complexities of distinguishing $\mathcal{P}_0^f$ and $\mathcal{P}_1^f$ and distinguishing $\mathcal{P}_0^g$ and $\mathcal{P}_1^g$ respectively. Then, the randomized p-parallel query complexity of distinguishing the distributions $\mathcal{P}_0 = \mathcal{P}_0^f \times \mathcal{P}_0^g$ and $\mathcal{P}_1 = \mathcal{P}_1^f \times \mathcal{P}_1^g$ is $\mathsf{R}^{p\|}(\mathcal{P}_0, \mathcal{P}_1) = \Omega(\min(r_f, r_g))$.*

Proof. Let $\mathcal{P}_{\text{hybrid}} = \mathcal{P}_0^f \times \mathcal{P}_1^g$. Then, since one can sample from the distribution $\mathcal{P}_0^f$ without any queries, distinguishing $\mathcal{P}_0$ from $\mathcal{P}_{\text{hybrid}}$ allows one to distinguish $\mathcal{P}_0^g$ from $\mathcal{P}_1^g$. Thus, $\mathsf{R}^{p\|}(\mathcal{P}_0, \mathcal{P}_{\text{hybrid}}) = \Omega(r_g)$. Similarly, $\mathsf{R}^{p\|}(\mathcal{P}_{\text{hybrid}}, \mathcal{P}_1) = \Omega(r_f)$. Suppose that a randomized p-parallel algorithm $\mathcal{A}$ making t queries could distinguish $\mathcal{P}_0$ and $\mathcal{P}_{\text{hybrid}}$ with probability $1/2 + \alpha(t)$ and distinguish $\mathcal{P}_{\text{hybrid}}$ and $\mathcal{P}_1$ with probability $1/2 + \beta(t)$. Then, $\mathcal{A}$ can distinguish $\mathcal{P}_0$ and $\mathcal{P}_1$ with probability at most $1/2 + \alpha(t) + \beta(t)$. Therefore, if we want $\alpha(t) + \beta(t) = \Omega(1)$, then we must choose t so that either $\alpha(t) = \Omega(1)$ or $\beta(t) = \Omega(1)$ (or both). It follows that $\mathsf{R}^{p\|}(\mathcal{P}_0, \mathcal{P}_1) = \min(\mathsf{R}^{p\|}(\mathcal{P}_0, \mathcal{P}_{\text{hybrid}}), \mathsf{R}^{p\|}(\mathcal{P}_{\text{hybrid}}, \mathcal{P}_1)) = \Omega(\min(r_f, r_g))$. □

We are now ready to prove the following relationship between the randomized p-parallel query complexities of f, g and $\textsc{cor}(f,g)$.

Lemma 38. *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be any (partial) Boolean functions. Then, $\mathsf{R}^{p\|}(\textsc{cor}(f,g)) = \Theta(\min(\mathsf{R}^{p\|}(f), \mathsf{R}^{p\|}(g))$.*

Proof. Similar to the deterministic case, computing f or g is sufficient to compute $\textsc{cor}(f,g)$. Thus, $\mathsf{R}^{p\|}((\textsc{cor}(f,g)) \leq \min(\mathsf{R}^{p\|}(f), \mathsf{R}^{p\|}(g))$.

Now, we argue for the lower bound. Let $\mathcal{P}_0^f$ and $\mathcal{P}_1^f$ be respective distributions over X_0 and X_1 such that distinguishing them requires any randomized p-parallel algorithm to make $\Omega(\mathsf{R}^{p\|}(f))$ queries. Similarly, let $\mathcal{P}_0^g$ and $\mathcal{P}_1^g$ be respective distributions over Y_0 and Y_1 such that distinguishing them requires any randomized p-parallel algorithm to make $\Omega(\mathsf{R}^{p\|}(g))$ queries. That is, $r_f = \mathsf{R}^{p\|}(\mathcal{P}_0^f, \mathcal{P}_1^f) = \Omega(\mathsf{R}^{p\|}(f))$ and $r_g = \mathsf{R}^{p\|}(\mathcal{P}_0^g, \mathcal{P}_1^g) = \Omega(\mathsf{R}^{p\|}(g))$. By Proposition 37, we know that the randomized p-parallel query complexity of distinguishing the distributions $\mathcal{P}_0 = \mathcal{P}_0^f \times \mathcal{P}_0^g$ and $\mathcal{P}_1 = \mathcal{P}_1^f \times \mathcal{P}_1^g$ is $\mathsf{R}^{p\|}(\mathcal{P}_0, \mathcal{P}_1) = \Omega(\min(r_f, r_g)) = \Omega(\min(\mathsf{R}^{p\|}(f), \mathsf{R}^{p\|}(g))$. But, since $\mathcal{P}_0$ is a distribution over $X_0 \times Y_0$ and $\mathcal{P}_1$ is a distribution over $X_1 \times Y_1$, the task of distinguishing $\mathcal{P}_0$ and $\mathcal{P}_1$ is at most as hard as computing $\textsc{cor}(f,g)$. It follows that $\mathsf{R}^{p\|}(\textsc{cor}(f,g)) \geq \mathsf{R}^{p\|}(\mathcal{P}_0, \mathcal{P}_1) = \Omega(\min(\mathsf{R}^{p\|}(f), \mathsf{R}^{p\|}(g))$. □

Last, we will prove a relationship between the quantum query complexity of f, g and $\textsc{cor}(f,g)$.

Lemma 39. *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be any (partial) Boolean functions on n_f and n_g bits respectively. Then, $\mathsf{Q}(\textsc{cor}(f,g)) = \Theta(\min(\mathsf{Q}(f), \mathsf{Q}(g))$.*

Proof. Similar to the deterministic and the randomized case, computing f or g is sufficient to compute $\textsc{cor}(f,g)$. Thus, $\mathsf{Q}(\textsc{cor}(f,g)) = O(\min(\mathsf{Q}(f), \mathsf{Q}(g)))$.

Now, we argue for the lower bound. Let $\Gamma^{(f)} \in \mathbf{R}^{|X_0|\times|X_1|}$ and $\Gamma^{(g)} \in \mathbf{R}^{|Y_0|\times|Y_1|}$ be an optimal adversary matrices for the functions f and g respectively. That is, $\mathsf{Adv}(f) = \frac{\left\|\Gamma^{(f)}\right\|}{\min_{i\in[n_f]}\left\|\Gamma_i^{(f)}\right\|}$ and $\mathsf{Adv}(g) = \frac{\left\|\Gamma^{(g)}\right\|}{\min_{i\in[n_f+1,n_g]}\left\|\Gamma_i^{(g)}\right\|}$. Let $\Gamma^{(\textsc{cor}(f,g))} = \Gamma(f) \otimes \Gamma(g) \in \mathbf{R}^{|X_0||Y_0|\times|X_1||Y_1|}$, where $\otimes$ refers to the Kronecker product. It is easy to see that $\Gamma^{(\textsc{cor}(f,g))}$ is an adversary matrix for $\textsc{cor}(f,g)$. Moreover, for any $i \in [n_f]$, $\Gamma_i^{(\textsc{cor}(f,g))} = \Gamma_i^{(f)} \otimes \Gamma^{(g)}$. Similarly, for any $i \in [n_f+1, n_g]$, $\Gamma_i^{(\textsc{cor}(f,g))} = \Gamma^{(f)} \otimes \Gamma_i^{(g)}$. Therefore,

$$\begin{aligned}
\mathsf{Q}(\textsc{cor}(f,g)) &= \Omega(\mathsf{Adv}(\textsc{cor}(f,g)) \\
&= \Omega\left(\min_{i\in[n_f+n_g]} \frac{\left\|\Gamma^{(\textsc{cor}(f,g))}\right\|}{\left\|\Gamma_i^{(\textsc{cor}(f,g))}\right\|}\right) \\
&= \Omega\left(\min\left(\min_{i\in[n_f]} \frac{\left\|\Gamma^{(f)}\right\|}{\left\|\Gamma_i^{(f))}\right\|}, \min_{i\in[n_f+1,n_g]} \frac{\left\|\Gamma^{(g)}\right\|}{\left\|\Gamma_i^{(g)}\right\|}\right)\right) \\
&= \Omega\left(\min\left(\mathsf{Adv}(f), \mathsf{Adv}(g)\right)\right) \\
&= \Omega(\min(\mathsf{Q}(f), \mathsf{Q}(g)))
\end{aligned}$$

□

For our applications, it is sufficient to consider the $p = 1$ case. Moreover, it is not clear how we would generalize the above proof for $p > 1$ since we will need to consider sets of indices that contains inputs to f and inputs to g. Therefore, we leave extending Lemma 39 to p-parallel quantum query complexity as future work.

5.2 Separations between randomized and deterministic complexities

We begin by showing an unbounded separation for partial functions between randomized and deterministic parallel query complexities, while maintaining no sequential separation.

5.2.1 Partial function separation

We will choose f to be the pointer chasing function (Problem 15) with instance size N and chain length $k = \sqrt{N}$ and g to be the balanced composition of PARITY and the DEUTSCH JOZSA MODIFIED function (Problem 13). Then, we define DEUTSCH-JOSZA ANA FUNCTION as the function $\textsc{cor}(f,g)$ (Problem 35). Precisely,

Problem 40 (DEUTSCH-JOSZA ANA)**.** *Let $N = 2^n$, $f = \textsc{pointer}_{N,\sqrt{N}}$ and $g = \bigoplus_{\sqrt{N}} \circ \textsc{dj}_{\sqrt{N}}$.*

Input: *Oracles for strings $X \in \{0,1\}^{Nn}$ and $Y \in \{0,1\}^N$.*
Promise: *Y satisfies the promise of DEUTSCH JOZSA MODIFIED (Problem 13), and $f(X) = g(Y)$.*
Question: *Decide $f(X) = g(Y)$.*

We will sometimes refer to the DEUTSCH-JOZSA ANA function on parameter N as djana_N.

Below, we show that the DEUTSCH-JOZSA ANA function satisfies our criteria of a parallel separation without a sequential separation for randomized vs deterministic complexities. In the sequential case, a strategy for both the deterministic and the randomized algorithm is to follow the pointer to compute DEUTSCH-JOZSA ANA in $\tilde{O}(\sqrt{N})$ queries, and lower bounds for f, g and $\textsc{cor}(f,g)$ show neither can do better. In the parallel case with $\tilde{\Theta}(\sqrt{N})$ parallelism, the randomized algorithm can solve g in 1 round because the PARITY component of g can be computed in parallel and DEUTSCH JOZSA MODIFIED requires 1 query. However, the deterministic algorithm is still forced to spend $\sqrt{N}$ queries for DEUSTSCH JOZSA MODIFIED even if it can solve parity in parallel. Formally, we prove the theorem:

Theorem 41. *Let h be the DEUTSCH-JOZSA ANA function as defined in Problem 40. Then for some $p = \tilde{\Theta}(\sqrt{N})$,*

$$\textit{(i)}\ \mathsf{D}(h) = O\left(\sqrt{N}\right), \quad \textit{(ii)}\ \mathsf{R}(h) = \Omega\left(\sqrt{N}\right), \quad \textit{(iii)}\ \mathsf{D}^{p\|}(h) = \tilde{\Omega}\left(\sqrt{N}\right), \quad \textit{(iv)}\ \mathsf{R}^{p\|}(h) = 1.$$

Proof. We prove each of the items one by one as follows.

(i) By Theorem 16, we know that $\mathsf{D}(f) = O(\sqrt{N})$. Since $\mathsf{D}(h) = \min(\mathsf{D}(f), \mathsf{D}(g))$ from Lemma 36, we get $\mathsf{D}(h) = O(\sqrt{N})$.

(ii) From Theorem 16, we have $\mathsf{R}(f) = \Omega(\sqrt{N})$ and from Theorem 19, we have $\mathsf{R}(g) = \Omega(\sqrt{N})$. It follows, from Lemma 38, that $\mathsf{R}(h) = \Omega(\min(\mathsf{R}(f), \mathsf{R}(g))) = \Omega(\sqrt{N})$.

(iii) From Theorem 16, we know that $\mathsf{D}^{p\|}(f) = \Omega(\sqrt{N})$ and from Theorem 19, we have that $\mathsf{D}(g) = \Omega(N)$ so $\mathsf{D}^{p\|}(g) = \Omega(N/p) = \tilde{\Omega}(\sqrt{N})$. Since $\mathsf{D}^{p\|}(h) = \min(\mathsf{D}^{p\|}(f), \mathsf{D}^{p\|}(g))$ from Lemma 36, we get $\mathsf{D}^{p\|}(h) = \tilde{\Omega}(\sqrt{N})$.

(iv) Since DEUTSCH JOZSA MODIFIED is defined to have randomized query complexity 1 and PARITY can be perfectly parallelized, we have $\mathsf{R}^{p\|}(g) = 1$. Solving g is sufficient to solve h so $\mathsf{R}^{p\|}(h) = 1$.

□

5.2.2 Total function separation

We totalize the partial function obtained in the previous section to show an unbounded total function parallel separation without sequential separation (between randomized and deterministic complexities).

Problem 42 (CHEATSHEET DEUTSCH-JOZSA ANA)**.** *Let $f : \{0,1\}^N \to \{0,1\}$ and $g : \{0,1\}^{N^2} \to \{0,1\}$ be defined as $f = \mathsf{djana}_N$ (see Problem 40) and $g = (\textsc{And} \circ \textsc{Or})_{N^2}$. Let $c = 10 \log N$.*

Input: *An oracle for a string $X \in \{0,1\}^M$ (with $M = \tilde{\Theta}(N^{12})$[‡‡]).*
Question: *Compute $(f \circ g)^c_{\mathsf{cs}}(X)$ (see Definition 20).*

Next, we show that this function demonstrates unbounded parallel advantage without sequential advantage.

Theorem 43. *Let h be the CHEATSHEET DEUTSCH-JOZSA ANA function as defined in Problem 42. Then for some $p = \Theta(N^{5/2} \log^3 N)$,*

[‡‡]Since a certificate of $\textsc{And} \circ \textsc{Or}$ on N^2 bits could be represented using $\tilde{\Theta}(N)$ bits, we will have the number of input bits for the function $(f \circ g)^c_{\mathsf{cs}}$ to be $\tilde{\Theta}(N^{12})$.

(i) $\mathsf{D}(h) = \tilde{O}\left(N^{5/2}\right)$, *(ii)* $\mathsf{R}(h) = \tilde{\Omega}\left(N^{5/2}\right)$, *(iii)* $\mathsf{D}^{p\|}(h) = \tilde{\Omega}\left(\sqrt{N}\right)$, *(iv)* $\mathsf{R}^{p\|}(h) \leq 3$.

Proof. We will prove each of the statements one by one. Recall that $f = \mathsf{djana}_N$ and $g = (\textsc{And} \circ \textsc{Or})_{N^2}$ in Problem 42.

(i) By part (i) of Theorem 41 and Theorem 19, we know that $\mathsf{D}(f \circ g) = O(N^{5/2})$. It follows, by part (i) of Proposition 23, that $\mathsf{D}(h) = \tilde{O}(N^{5/2})$.

(ii) By part (ii) of Theorem 41 and Theorem 19, we know that $\mathsf{R}(f \circ g) = \Omega(N^{5/2})$. It follows, by Theorem 22, that $\mathsf{R}(h) = \tilde{\Omega}(N^{5/2})$.

(iii) By part (iii) of Theorem 41, we know that $\mathsf{D}^{\lfloor p/N^2 \rfloor \|}(f) = \tilde{\Omega}(\sqrt{N})$. Since $\mathsf{D}(g) = \Theta(N^2)$, it follows, by Theorem 66, that $\mathsf{D}^{p\|}(f \circ g) = \Omega\left(\mathsf{D}^{\lfloor p/N^2 \rfloor \|}(f)\right) = \Omega(\sqrt{N})$. Therefore, using Theorem 71, we have that $\mathsf{D}^{p\|}(h) = \tilde{\Omega}(\sqrt{N})$.

(iv) The desired result directly follows from part (i) of Proposition 24.

□

The previous results are shown with an input of size $\tilde{\Theta}(N^{12})$. We state the corollary of the above theorem below using inputs of size M to make clear the exponents in the separation.

Corollary 44. *There exists a total Boolean function $h : \{0,1\}^M \to \{0,1\}$ and an integer $p \in [M]$ such that*

(i) $\mathsf{D}(h) = \tilde{O}\left(M^{5/24}\right)$, *(ii)* $\mathsf{R}(h) = \tilde{\Omega}\left(\mathsf{D}(h)\right)$, *(iii)* $\mathsf{D}^{p\|}(h) = \tilde{\Omega}\left(M^{1/24}\right)$, *(iv)* $\mathsf{R}^{p\|}(h) \leq 3$.

5.3 Separations between quantum and randomized complexities

Using the same framework, we can separate quantum and randomized parallel query complexities, while maintaining no sequential separation. We first give partial function separations, and then totalize them.

5.3.1 Partial function separation

We will choose f to be the pointer chasing function (Problem 15) with instance size N and chain length $k = N^{1/3}$ and g to be the composition of PARITY with instance size $N^{1/3}$ and the FORRELATION function (Problem 14) with instance size $N^{2/3}$. Then, we define FORRELATION ANA FUNCTION as the function $\textsc{cor}(f, g)$ (Problem 35). Precisely,

Problem 45 (FORRELATION ANA FUNCTION)**.** *Let $N = 2^n$, $f = \mathit{POINTER}_{N,N^{1/3}}$ and $g = \bigoplus_{N^{1/3}} \circ \mathit{FOR}_{N^{2/3}}$.*

Input: *Oracles for strings $X \in \{0,1\}^{Nn}$ and $Y \in \{0,1\}^N$.*
Promise: *Y satisfies the promise of FORRELATION (Problem 14), and $f(X) = g(Y)$.*
Question: *Decide $f(X) = g(Y)$.*

Below, we show that the FORRELATION ANA function satisfies our criteria of a parallel separation without a sequential separation for quantum vs randomized complexities. In the sequential case, a strategy for both the quantum and the randomized algorithm is to follow the pointer to compute FORRELATION ANA in $\tilde{O}(N^{1/3})$ queries, and lower bounds for f, g and $\textsc{cor}(f, g)$ show neither can do better. In the parallel case with $\tilde{\Theta}(N^{1/3})$ parallelism, the quantum algorithm can solve g in 1 query because the PARITY component of g can be computed in parallel and FORRELATION is takes just 1

query. However, the randomized algorithm is still forced to spend $N^{1/3}$ queries for FORRELATION even if it can solve parity in parallel. Formally, we prove the theorem:

Theorem 46. *Let h be the FORRELATION ANA FUNCTION as defined in Problem 45. Then for some $p = \tilde{\Theta}(N^{1/3})$,*

$$\textit{(i)}\ \mathsf{R}(h) = \tilde{O}\left(N^{1/3}\right), \quad \textit{(ii)}\ \mathsf{Q}(h) = \Omega\left(N^{1/3}\right), \quad \textit{(iii)}\ \mathsf{R}^{p\|}(h) = \tilde{\Omega}\left(N^{1/3}\right), \quad \textit{(iv)}\ \mathsf{Q}^{p\|}(h) = 1.$$

Proof. We prove each of the items one by one as follows.

(i) By Theorem 16, we know that $\mathsf{R}(f) = O(N^{1/3})$. Since $\mathsf{R}(h) = O(\min(\mathsf{R}(f), \mathsf{R}(g)))$ from Lemma 38, we get $\mathsf{R}(h) = O(N^{1/3})$.

(ii) From Theorem 16, we have $\mathsf{Q}(f) = \Omega(N^{1/3})$ and from Theorem 19, we have $\mathsf{Q}(g) = \Omega(N^{1/3})$. It follows, from Lemma 38, that $\mathsf{Q}(h) = \Omega(\min(\mathsf{Q}(f), \mathsf{Q}(g))) = \Omega(N^{1/3})$.

(iii) From Theorem 16, we know that $\mathsf{R}^{p\|}(f) = \Omega(N^{1/3})$ and from Theorem 19, we have that $\mathsf{R}(g) = \Omega(N^{1/3} \cdot \sqrt{N^{2/3}}) = \Omega(N^{2/3})$ so $\mathsf{R}^{p\|}(g) = \Omega(N^{2/3}/p) = \tilde{\Omega}(N^{1/3})$. Since $\mathsf{R}^{p\|}(h) = \Omega(\min(\mathsf{R}^{p\|}(f), \mathsf{R}^{p\|}(g)))$ from Lemma 36, we get $\mathsf{R}^{p\|}(h) = \tilde{\Omega}(N^{1/3})$.

(iv) Since FORRELATION has quantum query complexity 1 and PARITY can be perfectly parallelized, we have $\mathsf{Q}^{p\|}(g) = 1$. Solving g is sufficient to solve h so $\mathsf{Q}^{p\|}(h) = 1$. □

5.3.2 Total function separation

We totalize the partial function obtained in the previous section to show an unbounded total function parallel separation without sequential separation (between quantum and randomized complexities).

Problem 47 (CHEATSHEET FORRELATION ANA)**.** *Let $f : \{0,1\}^N \to \{0,1\}$ and $g : \{0,1\}^{N^2} \to \{0,1\}$ be defined as $f = \mathsf{forana}_N$ (see Problem 45) and $g = BKK_{N^2}$ (see Problem 11). Let $c = 10 \log N$.*

Input: *An oracle for a string $X \in \{0,1\}^M$ (with $M = \tilde{\Theta}(N^{12})$[§§]).*
Question: *Compute $(f \circ g)^c_{\mathsf{cs}}(X)$ (see Definition 20).*

We use the above function to prove an unbounded advantage in parallel query complexities but no sequential advantage between randomized and quantum algorithms.

Theorem 48. *Let h be the CHEATSHEET FORRELATION ANA function as defined in Problem 47. Then for some $p = \Theta(N^{7/3} \log^3 N)$,*

$$\textit{(i)}\ \mathsf{R}(h) = \tilde{O}\left(N^{7/3}\right), \quad \textit{(ii)}\ \mathsf{Q}(h) = \tilde{\Omega}\left(N^{7/3}\right), \quad \textit{(iii)}\ \mathsf{R}^{p\|}(h) = \tilde{\Omega}\left(N^{1/3}\right), \quad \textit{(iv)}\ \mathsf{Q}^{p\|}(h) \leq 3.$$

Proof. We will proof each of the statements one by one. Recall that $f = \mathsf{djana}_N$ and $g = (\text{AND}\circ\text{OR})_{N^2}$ in Problem 42.

(i) By part (i) of Theorem 46 and Theorem 19, we know that $\mathsf{R}(f \circ g) = O(N^{7/3})$. It follows, by part (ii) of Proposition 23, that $\mathsf{R}(h) = \tilde{O}(N^{7/3})$.

(ii) We know, from Theorem 12, that $\mathsf{Q}(g) = \tilde{\Omega}(N^2)$. By part (ii) of Theorem 46 and Theorem 19, we have that $\mathsf{Q}(f \circ g) = \tilde{\Omega}(N^{7/3})$. It follows, by Theorem 22, that $\mathsf{Q}(h) = \tilde{\Omega}(N^{7/3})$.

[§§]Since a certificate of AND ∘ OR on N^2 bits could be represented using $\tilde{\Theta}(N)$ bits, we will have the number of input bits for the function $(f \circ g)^c_{\mathsf{cs}}$ to be $\tilde{\Theta}(N^{12})$.

(iii) By part (iii) of Theorem 46, we know that $\mathsf{R}^{\lfloor p/N^2 \rfloor \parallel}(f) = \Omega(N^{1/3})$. It follows, by Theorem 69, that $\mathsf{R}^{p\parallel}(f \circ g) = \Omega\left(\mathsf{R}^{\lfloor p/N^2 \rfloor \parallel}(f)\right) = \tilde{\Omega}(N^{1/3})$. Therefore, using Theorem 73, we have that $\mathsf{R}^{p\parallel}(h) = \tilde{\Omega}(N^{1/3})$.

(iv) The desired result directly follows from part (ii) of Proposition 24.

□

The above theorem was proven for a function using input size $\tilde{\Theta}(N^{12})$. We restate it as a corollary below using input size M to make clear the size of the exponents in the separation.

Corollary 49. *There exists a total Boolean function $h : \{0,1\}^M \to \{0,1\}$ and an integer $p \in [M]$ such that*

(i) $\mathsf{R}(h) = \tilde{O}\left(M^{7/36}\right)$, *(ii)* $\mathsf{Q}(h) = \tilde{\Omega}\left(\mathsf{D}(h)\right)$, *(iii)* $\mathsf{R}^{p\parallel}(h) = \tilde{\Omega}\left(M^{1/36}\right)$, *(iv)* $\mathsf{Q}^{p\parallel}(h) \leq 3$.

6 Separations with two layers of adaptivity

So far we have largely been dealing with separations in the depth (or number of layers or rounds) of an algorithm given a fixed width (or parallelism), that is, separations in $\mathsf{D}^{p\parallel}(f)$, $\mathsf{R}^{p\parallel}(f)$ and $\mathsf{Q}^{p\parallel}(f)$. The relevant dual question to ask is given a fixed depth k, what total function separation is possible for the widths, denoted as $\mathsf{D}^{k\perp}(f)$, $\mathsf{R}^{k\perp}(f)$ and $\mathsf{Q}^{k\perp}(f)$? Moreover, what is the least depth k for which a polynomial separation is possible between the widths? From [Mon10], we know that for $k = 1$, for all total f, there can be at most a constant factor of separation, that is $\mathsf{D}^{1\perp}(f) = \Theta(\mathsf{R}^{1\perp}(f)) = \Theta(\mathsf{Q}^{1\perp}(f))$. From Theorem 30 and Theorem 27 in this work, we know that at $k = 3$, there is a polynomial separation. In particular, there exist functions f and g such that $\mathsf{R}^{3\perp}(f) = \tilde{O}\left(D^{3\perp}(f)^{1-\delta}\right)$ and $\mathsf{Q}^{3\perp}(g) = \tilde{O}\left(\mathsf{R}^{3\perp}(g)^{1-\delta'}\right)$ for some $\delta, \delta' > 0$. Thus, the pertinent question is: is it possible to have a polynomial separation for total functions at $k = 2$? In this section we show that it is indeed possible—two layers of adaptivity are sufficient for a polynomial separation in parallelism or width between deterministic and randomized, and randomized and quantum algorithms. More generally, we show:

Theorem 50. *(Informal) Any partial function f that shows a polynomial separation in parallelism with one layer of adaptivity (R vs D or Q vs R) can be converted into a total function h that shows a polynomial separation in parallelism with two layers of adaptivity.*

Before showing how we can obtain h given f, we first give some intuition. We discuss here the Q vs R separation but the same idea holds for R vs D. Let's start with the canonical cheat sheet function (Definition 21) used to show the separations in Theorem 30 and Theorem 27. The separation here requires three rounds of adaptivity. At a high level, the upper bound for Q is given by doing the following in the three rounds:

1. Solve the hard partial function to obtain an address.
2. Read the certificate at the address in the cheat sheet.
3. Use the certificate to go back and verify whether the input was in the domain of the hard partial function.

R is unable to solve the hard partial function and obtain the location of the cheat sheet in the same amount of parallelism, hence the separation.

Our challenge is to compress the above into two rounds. To achieve this, we remove the data dependence from step (1) to step (2). In particular, we will give the certificates of the hard partial function in a given fixed input position. However, we would still like to force the algorithm to solve the problem before (or in parallel to) reading the certificates: to accomplish this, we will have a third "data" section of the input. The output of the function will depend on a certain bit of the data register, specifically the one pointed to by the output of the hard partial function. In this way, an algorithm capable of solving the hard partial function without the certificates can find the data bit in the second round of queries. However, an algorithm that needs the certificates will not be able to learn the data bit until the third round. This function is still total, because the certificates can—also in the second round—be used to verify that the partial function input is from the domain.

Note a subtle point in the prior intuition: an algorithm can learn a certificate of the hard partial function before it solves said function, in seeming contrast to the cheatsheet construction. If the *form* of the certificate reveals the output of the function (e.g. for $\textsc{And} \circ \textsc{Or}$, a minimal YES certificate points to a single position in each block, whereas a minimal NO certificate points to a full block), then the algorithm doesn't need to actually solve the function in the first round. We avoid this using so-called "bi-certificates" for the $\textsc{And} \circ \textsc{Or}$ function. These are pointers to both an entire block, and a single index in each block, such that either a YES or NO instance could be certified. In this way, even after learning all the bi-certificates, an algorithm must still query the hard partial function to learn its output.

We now define the function h given the partial function f. Then, in the following sections, we use h to show separations between D and R, and R and Q.

Problem 51. *2-Adaptive-F Let $f : \{0,1\}^N \to \{0,1\}$ be a partial function. The input bit string to $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ is divided into three components:*

1. *ADD: The address component of size $= \tilde{\Theta}(N^3)$*
2. *BC: The bicertificate component of size $= \tilde{\Theta}(N^2)$*
3. *DT: The data component of size $= N^3$*

We then make the following indexing/ definitions:

1. *Divide ADD into $3 \log N$ segments of size N^3 each, calling each segment $\textsc{add}[i]$. Further divide each segment into N sub segments of size N^2 each, calling each sub segment $\textsc{add}[i,j]$. Each sub segment is further divided into N sized blocks $\textsc{add}[i,j,k]$.*
2. *Similarly, divide BC into $3 \log N$ segments of size $\tilde{\Theta}(N^2)$ each, calling each segment $\textsc{bc}[i]$. Further divide each segment into N sub segments of size $\tilde{\Theta}(N)$[*] calling each sub segment $\textsc{bc}[i,j]$. $\textsc{bc}[i,j]$ is called the "bicertificate" corresponding to $\textsc{add}[i,j]$.*
3. *Define the "f-input bits" $\textsc{in}[i,j] = \textsc{And} \circ \textsc{Or}(\textsc{add}[i,j])$, and the "f-input" $\textsc{in}[i] = \textsc{in}[i,0] \frown \textsc{in}[i,1] \frown \textsc{in}[i,2] \ldots \textsc{in}[i,N-1]$*
4. *If each $\textsc{in}[i]$ is in the domain of f, define the " target address bits" $\textsc{tg}[i] = f(\textsc{in}[i])$ and the "target address" $\textsc{tg} = \textsc{tg}[0] \frown \textsc{tg}[1] \frown \textsc{tg}[2] \ldots \textsc{tg}[3 \log N - 1]$*
5. *A valid bicertificate is a set of locations that could form a zero-certificate concatenated with a set of locations that could form a one-certificate (the actual instance is not taken into account).*

[*]The $\tilde{\Theta}$ hides a log and a constant factor. It comes from the number of bits required to specify $2N$ locations of the ADD bit string.

In the case of AND ∘ OR*, a valid bicertificate* BC$[i,j]$ *for* ADD$[i,j]$ *is locations corresponding to any block* ADD$[i,j,k]$ *(the zero certificate) concatenated with a set of one location in* ADD$[i,j,k]$ *for each* k *(the one certificate).*

6. *The intersection point* IP$[i,j]$ *defined for a valid* AND ∘ OR *bicertificate* BC$[i,j]$ *refers to the single location that is part of both the zero- and the one- certificate.*

Then, a given input is a YES instance to h *if and only if:*

1. *For all* $i \in [0...3\log N - 1]$ *and* $j \in [0...N-1]$*, we have* BC$[i,j]$ *is a valid* AND ∘ OR*,*
2. *Every* BC$[i,j]$ *certifies the output of* AND ∘ OR(ADD$[i,j]$)*,*
3. *All* IN$[i]$ *are within the domain of* f*, and*
4. DT$[$TG$]=1$

We can now use 2-ADAPTIVE-F (depicted in Figure 4) to show the desired separations. In Section 6.1, we show the deterministic vs randomized separation and in Section 6.2, we show the randomized vs quantum separation.

6.1 Separation between randomized and deterministic complexities

Theorem 52. *Let* $f : \mathcal{D} \to \{0,1\}$ *for* $\mathcal{D} \subset \{0,1\}^N$ *be a partial function that satisfies the following where* $0 \le s, l \le 1$*:*

$$\mathsf{R}^{1\perp}(f) = \tilde{O}(N^s) \qquad \mathsf{D}^{1\perp}(f) = \tilde{\Omega}(N^l)$$

Then $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ *as constructed in Problem 51 is a total function that satisfies:*

$$\mathsf{R}^{2\perp}(h) = \tilde{O}(N^{s+2}) \qquad \mathsf{D}^{2\perp}(h) = \tilde{\Omega}(N^{l+2})$$

Proof. The statement follows from the deterministic lower bound proved in Lemma 53 and the randomized upper bound proved in Lemma 54.

□

By taking f as the DEUTSCH JOZSA problem, for which $\mathsf{R}^{1\perp}(f) = k$ where k is constant, and $\mathsf{D}^{1\perp}(f) = \tilde{\Omega}(N)$, we have the desired polynomial separation.

Lemma 53. *Let* $f : \mathcal{D} \to \{0,1\}$ *for* $\mathcal{D} \subset \{0,1\}^N$ *be a partial function that satisfies* $\mathsf{D}^{1\perp}(f) = \tilde{\Omega}(N^l)$ *where* $0 \le l \le 1$*. Then* $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ *as constructed in Problem 51 is a total function that satisfies* $\mathsf{D}^{2\perp}(h) = \tilde{\Omega}(N^{l+2})$*.*

Proof. It suffices to show that for any 2 round deterministic algorithm that makes $\tilde{O}\left(N^{l+2-\epsilon}\right)$ queries for any $\epsilon > 0$, it is always possible to adversarially answer the queries such that the algorithm fails.

Round 1 The algorithm submits $\tilde{O}(N^{2+l-\epsilon})$ queries each to ADD, BC and DT. Answer the queries using the following strategy:

1. For each segment ADD$[i]$, the algorithm can fully query at most $\tilde{O}(N^{2+l-\epsilon}/N^2) = \tilde{O}(N^{l-\epsilon})$ ADD$[i,j]$'s, and there remain $\tilde{\Omega}(N^{\epsilon})$ ADD$[i,j]$'s with at least one bit unqueried.

(a) For $\textsc{add}[i,j]$ with at least one unqueried bit: Select a valid bicertificate $\textsc{bc}[i,j]$ such that the intersection point $\textsc{ip}[i,j]$ points to at an unqueried bit. For the queried bits, if l belongs to the one-certificate of $\textsc{bc}[i,j]$, set $\textsc{add}[i,j](l) = 1$, else set $\textsc{add}[i,j](l) = 0$. This ensures that both $\textsc{And} \circ \textsc{Or}(\textsc{add}[i,j]) = 1$ and $\textsc{And} \circ \textsc{Or}(\textsc{add}[i,j]) = 0$ are still consistent with answered queries.

(b) For the $\textsc{add}[i,j]$ which are fully queried: There are at most $\tilde{O}(N^{l-\epsilon})$ such $\textsc{add}[i,j]$'s, therefore this fixes at most $\tilde{O}(N^{l-\epsilon})$ bits of $\textsc{in}[i]$. Answer the queries to the $\textsc{add}[i,j]$'s such that $\textsc{in}[i]$ can be completed to both a 0-instance and a 1-instance of f. This should always be possible, else it contradicts $\mathsf{D}^{1\perp}(f) = \tilde{\Omega}(N^l)$.

2. For queries to $\textsc{dt}$, answer anything.

Round 2 At this stage, note that the algorithm can fully learn $\textsc{bc}$. The algorithm once again submits $\tilde{O}(N^{2+l-\epsilon})$ queries to $\textsc{add}$, $\textsc{bc}$ and $\textsc{dt}$. Answer the queries using the following strategy:

1. Designate any location of $\textsc{dt}$ that is unqueried in both rounds as the target address t'. $\|\textsc{dt}\| = N^3$ and only $\tilde{O}(N^{2+l-\epsilon})$ queries have been made to it, so there should be several such locations.
2. Setting $\textsc{tg} = t'$, for each i, select an $\textsc{in}[i]$ consistent with the queries such that $f(\textsc{in}[i]) = \textsc{tg}[i]$. In the first round we ensured this is always possible. Then answer queries to the $\textsc{add}[i,j]$ such that $\textsc{And} \circ \textsc{Or}(\textsc{add}[i,j]) = \textsc{in}[i,j]$.
3. For queries to $\textsc{dt}$, answer anything.

After Round 2, as the target location has not been queried, both 0 and 1 outputs of h are consistent with the queries made. Hence the deterministic algorithm fails, proving the claim. □

Lemma 54. *Let $f : \mathcal{D} \to \{0,1\}$ for $\mathcal{D} \subset \{0,1\}^N$ be a partial function that satisfies $\mathsf{R}^{1\perp}(f) = \tilde{O}(N^s)$ where $0 \le s \le 1$. Then $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ as constructed in Problem 51 is a total function that satisfies $\mathsf{R}^{2\perp}(h) = \tilde{O}(N^{s+2})$.*

Proof. The following randomized algorithm proves the statement.

Round 1 We have $\mathsf{R}^{1\perp}(f) = \tilde{O}(N^s)$. Therefore, by composition, $\mathsf{R}^{1\perp}(f \circ \textsc{And} \circ \textsc{Or}) = \tilde{O}(N^{2+s})$. Thus in $\tilde{O}(N^{s+2})$ queries the algorithm can determine each $\textsc{tg}[i] := f \circ \textsc{And} \circ \textsc{Or}(\textsc{add}[i])$ with probability $1/2 + c$ for any constant c. By repeating the algorithm non-adaptively $O(\log N)$ times and taking majority vote for each $\textsc{tg}[i]$, the algorithm can determine $\textsc{tg}$ with constant probability c'. The algorithm can also query the full $\textsc{bc}$ and check condition (1) in Problem 51.

Round 2 The algorithm queries $\textsc{add}$ at locations learned from $\textsc{bc}$ to verify conditions (2) and (3) of Problem 51. It also queries $\textsc{dt}$ at the $\textsc{tg}$ determined in Round 1. If conditions 1, 2 and 3 are satisfied, it answers with $\textsc{dt}[\textsc{tg}]$, else it answers 0. With probability c' we have $\textsc{dt}[\textsc{tg}]$ is correct, else it is a uniformly random bit. Hence the algorithm succeeds with probability $\ge \frac{1}{2} + c'$. □

6.2 Separation between quantum and randomized complexities

Theorem 55. *Let $f : \mathcal{D} \to \{0,1\}$ for $\mathcal{D} \subset \{0,1\}^N$ be a partial function that satisfies the following where $0 \le s, l \le 1$:*

$$\mathsf{Q}^{1\perp}(f) = \tilde{O}(N^s) \qquad\qquad \mathsf{R}^{1\perp}(f) = \tilde{\Omega}(N^l)$$

Then $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ *as constructed in Problem 51 is a total function that satisfies*

$$\mathsf{Q}^{2\perp}(h) = \tilde{O}(N^{s+2}) \qquad \mathsf{R}^{2\perp}(h) = \tilde{\Omega}(N^{l+2-o(1)})$$

Proof. The statement follows from the randomized lower bound proved in Lemma 56 and the quantum upper bound proved in Lemma 57. □

By taking f as the FORRELATION problem, for which $\mathsf{Q}^{1\perp}(f) = k$ where k is constant, and $\mathsf{R}^{1\perp}(f) = \tilde{\Omega}(\sqrt{N})$, we have the desired polynomial separation.

Lemma 56. *Let* $f : \mathcal{D} \to \{0,1\}$ *for* $\mathcal{D} \subset \{0,1\}^N$ *be a partial function that satisfies* $\mathsf{R}^{1\perp}(f) = \tilde{\Omega}(N^l)$ *where* $0 \leq l \leq 1$. *Then* $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ *as constructed in Problem 51 is a total function that satisfies* $\mathsf{R}^{2\perp}(h) = \tilde{\Omega}(N^{l+2-o(1)})$.

Proof. Using Yao's Lemma, observe that it is sufficient to show that there exists a distribution over inputs to h such that any deterministic algorithm making $\tilde{O}(N^{l+2-\epsilon})$ queries in 2 rounds for any $\epsilon > 0$ can succeed with probabilility at most $1/2 + o(1)$ over the distribution. We construct such a distribution below:

1. For $i \in [0 \ldots 3\log N - 1]$, select $\textsc{in}[i]$ from a hard distribution for f.
2. For each $\textsc{bc}[i,j]$, select valid bicertificates for $\textsc{And} \circ \textsc{Or}$ uniformly at random.
3. Fixing each $\textsc{in}[i]$ and $\textsc{bc}[i,j]$ as above fixes all the $\textsc{add}[i,j]$ as follows. For each bit location l of $\textsc{add}[i,j]$,
   (a) If $l = \textsc{ip}[i,j]$: set $\textsc{add}[i,j](l) = \textsc{in}[i,j]$
   (b) Else if l belongs to the one-certificate of $\textsc{bc}[i,j]$: set $\textsc{add}[i,j](l) = 1$
   (c) Else: set $\textsc{add}[i,j](l) = 0$
4. For $\textsc{dt}$, select a bitstring of length N^3 uniformly at random.

Round 1 The deterministic algorithm submits $\tilde{O}(N^{l+2-\epsilon})$ queries to $\textsc{add}$, $\textsc{bc}$ and $\textsc{dt}$. Thus after the first round, it learns $\tilde{O}(N^{l+2-\epsilon})$ bits in each $\textsc{add}[i]$, all of $\textsc{bc}$, and $\tilde{O}(N^{l+2-\epsilon})$ bits of $\textsc{dt}$. We will show that at this stage, among the inputs consistent with the queries, the probability distribution over all target addresses is almost uniform. That is $\Pr[\textsc{tg} = t'] \leq \frac{2}{N^3}$ for all $t' \in [0, N^3 - 1]$.

Recall that given $\textsc{bc}[i,j]$, every bit of $\textsc{add}[i,j]$ is fixed except $\textsc{add}[i,j](\textsc{ip}[i,j])$ which is equal to $\textsc{in}[i,j]$. Thus a query to $\textsc{add}[i,j](\textsc{ip}[i,j])$ fixes $\textsc{in}[i,j]$ whereas a query to any $\textsc{add}[i,j](l)$ where $l \neq \textsc{ip}[i,j]$ does not learn anything more about $\textsc{in}[i,j]$. The number of queries to $\textsc{add}[i]$ is $\tilde{O}(N^{l+2-\epsilon})$, and each of the contained N $\textsc{add}[i,j]$'s of length N^2 have one intersection point placed uniformly at random. Thus, by linearity of expectation, the expected number of intersection points queried are $\tilde{O}\left(\frac{N^{l+2-\epsilon}}{N^2}\right) = \tilde{O}(N^{l-\epsilon})$. Note also that the probability that any given intersection point is queried is independent of whether any other is queried: this follows from their independent uniform random placement. Therefore, using Markov's inequality,

$$\Pr\left[\#\text{Intersection points queried in } \textsc{add}[i] \geq N^{l-\epsilon/2}\right] = \tilde{O}\left(\frac{1}{N^{\epsilon/2}}\right) \tag{2}$$

Thus with the remaining probability $1-\tilde{O}\left(\frac{1}{N^{\epsilon/2}}\right)$, the number of intersection points queried in $\textsc{add}[i]$ is $< N^{l-\epsilon/2}$, or in other words the number of bits of $\textsc{in}[i]$ learned is $< N^{l-\epsilon/2}$. Thus, we have the probability of guessing $\textsc{tg}[i] := f(\textsc{in}[i])$ is

$$\begin{aligned}\Pr\left[\frac{\text{Determining } \textsc{tg}[i]}{< N^{l-\epsilon/2} \text{ bits of } \textsc{in}[i] \text{ known}}\right] &\leq \frac{1}{2} + \sqrt{\frac{\#\text{bits learned of } \textsc{in}[i]}{\mathsf{R}^{1\perp}(f)}} \\ &= \frac{1}{2} + \sqrt{\tilde{O}\left(\frac{N^{l-\epsilon/2}}{N^l}\right)} \\ &= \frac{1}{2} + \tilde{O}\left(\frac{1}{N^{\epsilon/4}}\right).\end{aligned} \tag{3}$$

where the first inequality must be true by the following argument, where we recall that the location of the input bits learned to f are independently, uniformly random. If it did not hold, then by repeating the process $\frac{\mathsf{R}^{1\perp}(f)}{\#\text{bits learned of } \textsc{in}[i]}$ times, f can be computed with $1/2 + const.$ probability in a total number of queries $< R^{1\perp}(f)$ which is a contradiction. Thus from Equation (2) and Equation (3), we have

$$\begin{aligned}\Pr[\text{Determining } \textsc{tg}[i]] &\leq \tilde{O}\left(\frac{1}{N^{\epsilon/2}}\right)(1) + \left(1 - \tilde{O}\left(\frac{1}{N^{\epsilon/2}}\right)\right)\left(\frac{1}{2} + \tilde{O}\left(\frac{1}{N^{\epsilon/4}}\right)\right) \\ &= \frac{1}{2} + \tilde{O}\left(\frac{1}{N^{\epsilon/4}}\right)\end{aligned} \tag{4}$$

Thus, for all $t' \in [0 \ldots N^3 - 1]$ and large enough N and any constant $\epsilon > 0$,

$$\Pr[\textsc{tg} = t'] \leq \left(\frac{1}{2} + \tilde{O}\left(\frac{1}{N^{\epsilon/4}}\right)\right)^{3\log N} \leq \frac{2}{N^3} \tag{5}$$

Round 2 Once again, the deterministic algorithm submits $\tilde{O}(N^{l+2-\epsilon})$ queries to $\textsc{add}$, $\textsc{bc}$ and $\textsc{dt}$. Let S be the set of all of the queries made to $\textsc{dt}$, both in the first round and the second round. We know that $|S| = \tilde{O}(N^{l+2-\epsilon})$. By the union bound, we have

$$\Pr[t' \in S] \leq \tilde{O}(N^{l+2-\epsilon})\left(\frac{2}{N^3}\right) = \tilde{O}\left(\frac{1}{N^{\epsilon}}\right). \tag{6}$$

Note that the first inequality holds for any constant $\epsilon > 0$, but not $\epsilon = 0$, as in this case Equation (5) fails to hold. Because $\textsc{dt}$ is selected uniformly at random, if t' is not queried, h can be answered with only probability $1/2$. Thus, the probability of success over this distribution for the deterministic algorithm is $\frac{1}{2} + \tilde{O}\left(\frac{1}{N^{\epsilon}}\right) = \frac{1}{2} + o(1)$ as desired. This proves the statement.

□

Lemma 57. *Let $f : \mathcal{D} \to \{0,1\}$ for $\mathcal{D} \subset \{0,1\}^N$ be a partial function that satisfies $\mathsf{Q}^{1\perp}(f) = \tilde{O}(N^s)$ where $0 \leq s \leq 1$. Then $h : \{0,1\}^{\tilde{\Theta}(N^3)} \to \{0,1\}$ as constructed in Problem 51 is a total function that satisfies $\mathsf{Q}^{2\perp}(h) = \tilde{O}(N^{s+2})$.*

Proof. The following quantum algorithm proves the statement:

Round 1 We have $\mathsf{Q}^{1\perp}(f) = \tilde{O}(N^s)$. Therefore, by composition, $\mathsf{Q}^{1\perp}(f \circ \textsc{And} \circ \textsc{Or}) = \tilde{O}(N^{s+2})$. Thus in $\tilde{O}(N^{s+2})$ queries the algorithm can determine each $\textsc{tg}[i] := f \circ \textsc{And} \circ \textsc{Or}(\textsc{add}[i])$ with probability $1/2 + c$ for any constant c. By repeating the algorithm non-adaptively $O(\log N)$ times and taking majority vote for each $\textsc{tg}[i]$, the algorithm can determine $\textsc{tg}$ with constant probability c'. The algorithm can also query the full $\textsc{bc}$ and check condition (1) in Problem 51.

Round 2 The algorithm queries $\textsc{add}$ at locations learned from $\textsc{bc}$ to verify conditions (2) and (3) of Problem 51. It also queries $\textsc{dt}$ at the $\textsc{tg}$ determined in Round 1. If conditions 1, 2 and 3 are satisfied, it answers with $\textsc{dt}[\textsc{tg}]$, else it answers 0. With probability c' we have $\textsc{dt}[\textsc{tg}]$ is correct, else it is a uniformly random bit. Hence the algorithm succeeds with probability $\geq \frac{1}{2} + c'$. □

7 A parallel quantum lower bound framework and applications

In this section we prove a modified version of the parallel quantum adversary theorem which is weaker, yet easy to work with, giving explicit lower bounds for read once formulas and symmetric functions. This theorem allows one to derive parallel lower bounds from sequential upper bounds. To motivate this result, we first give a barrier result for applying the parallel combinatorial adversary for lower bounding the parallel query complexity of read once formulas.

7.1 A parallel combinatorial adversary method barrier

In this section, we exhibit a general certificate barrier for the parallel combinatorial adversary method (Theorem 18), which in particular implies that the method cannot give a tight lower bound for the simple two layer $\textsc{And} \circ \textsc{Or}$ tree. Notably, in the sequential case the combinatorial adversary method suffices to give a tight lower bound [BS04]. This motivates our lower bound method, which overcomes this barrier while still being straightforward to apply. In particular, we show a lower bound for read-once formulas in Section 7.2, of which a tight lower bound for the two layer $\textsc{And} \circ \textsc{Or}$ tree is a special case. The barrier is described in the following theorem, and can be seen as a parallel analog of a theorem by Zhang ([Zha04], Theorem 10). Following the terminology therein, we use the term $Alb_{p\|}$ to refer to the parallel combinatorial adversary bound.

Theorem 58. *Let $f : \{0,1\}^N \to \{0,1\}$ be a total function, and let $Alb_{p\|}(f), C_0(f)$ and $C_1(f)$ refer to the parallel combinatorial adversary method bound (Theorem 18), the 0-certificate complexity and the 1-certificate complexity of f respectively. Then, we have $Alb_{p\|}(f) \leq \sqrt{\left\lceil \frac{C_0}{p} \right\rceil \left\lceil \frac{C_1}{p} \right\rceil}$.*

Proof. Let $X, Y, R, S, w(x,y), w_x, w_{x,S}, w_y$ and $w_{y,S}$ be defined as in Theorem 18. Thus,

$$Alb_{p\|}(f) = \max_{X,Y,R} \sqrt{\left(\frac{\min_x w_x}{\max_{x,S} w_{x,S}}\right)\left(\frac{\min_y w_y}{\max_{y,S'} w_{y,S'}}\right)}$$
$$\leq \max_{X,Y,R} \sqrt{\min_{x,S}\left(\frac{w_x}{w_{x,S}}\right) \min_{y,S'}\left(\frac{w_y}{w_{y,S'}}\right)}.$$

Therefore, to prove the statement, it is sufficient to show that for any choice of X, Y, R there exists an x, y, S, S' such that $\frac{w_x w_y}{w_{x,S} w_{y,S'}} \leq \left\lceil \frac{C_0(f)}{p} \right\rceil \left\lceil \frac{C_1(f)}{p} \right\rceil$. Let us assume this is false. That is for some

choice of X, Y, R,

$$\forall x, y, S, S', \quad w_x w_y > \left\lceil \frac{C_0(f)}{p} \right\rceil \left\lceil \frac{C_1(f)}{p} \right\rceil w_{x,S}\ w_{y,S'} \tag{7}$$

Let $C_{x,p}$ be a partition of the set of indices belonging to any certificate of x consisting of sets of size p, as well as one set of size $\leq p$ for the remaining indices. We have $|C_{x,p}| \leq \left\lceil \frac{C_0(f)}{p} \right\rceil$. Define $C_{y,p}$ likewise, and we have $|C_{y,p}| \leq \left\lceil \frac{C_1(f)}{p} \right\rceil$. Thus, Equation (7) implies $\forall\ x, y$

$$w_x w_y > \sum_{S \in C_{x,p}} w_{x,S} \sum_{S' \in C_{y,p}} w_{y,S'}.$$

Summing over x and y, we have

$$\begin{aligned} \sum_x w_x \sum_y w_y &> \sum_{x, S \in C_{x,p}} w_{x,S} \sum_{y, S' \in C_{y,p}} w_{y,S'} \\ &= \sum_{x,y} \left(\sum_{S \in C_{x,p} \mid x_S \neq y_S} w(x,y) \right) . \sum_{x,y} \left(\sum_{S' \in C_{y,p} \mid x_{S'} \neq y_{S'}} w(x,y) \right) \end{aligned} \tag{8}$$

Now because f is a total function, for each x, y there must exist some $S \in C_{x,p}$ such that $x_S \neq y_S$. Likewise, there must exist some $S' \in C_{y,p}$ such that $x_{S'} \neq y_{S'}$. Thus, for each x, y, the summation over S and S' in Equation (8) contains at least one term. Thus,

$$\begin{aligned} \sum_x w_x \sum_y w_y &> \sum_{x,y} w(x,y) . \sum_{x,y} w(x,y) \\ &> \sum_x w_x \sum_y w_y \end{aligned}$$

which is a contradiction. Hence, the statement follows. □

Corollary 59. *The parallel combinatorial adversary method Theorem 18 cannot provide a tight lower bound for* AND ∘ OR.

Proof. The AND ∘ OR function is a special case of a read once formula, so by Theorem 63 we have $\mathsf{Q}^{p\|}(\text{AND} \circ \text{OR}) = \Omega\left(\sqrt{\frac{N}{p}}\right)$. Furthermore, $\mathsf{Q}^{p\|}(\text{AND} \circ \text{OR}) = \tilde{O}\left(\sqrt{\frac{N}{p}}\right)$ by parallel Grover search, so this is tight up to log factors in N, p. However, we have $C_0(\text{AND} \circ \text{OR}) = C_1(\text{AND} \circ \text{OR}) = \sqrt{N}$. Thus, by Theorem 58, we have $Alb_{p\|}(\text{AND} \circ \text{OR}) \leq \left\lceil \frac{\sqrt{N}}{p} \right\rceil$. □

We note that the above proof only acts as a barrier for the parallel combinatorial adversarial method, but not the parallel version of the general positive weighted adversary method [Amb02, Zha04]. However, it is unclear how to assign weights using the general method to prove parallel lower bounds for read once formulas.

7.2 Lower bounds from nearest neighbor adversaries

For any total $f : \{0,1\}^N \to \{0,1\}$, call Γ^{NN} a nearest neighbor adversary matrix of f if it is an adversary matrix that only places non-zero weight on pairs that differ at a single entry. Formally:

$$\begin{aligned} \Gamma^{\mathsf{NN}}_{xy} &= \Gamma^{\mathsf{NN}}_{yx} \\ f(x) = f(y) &\Rightarrow \Gamma^{\mathsf{NN}}_{xy} = 0 \\ d(x,y) > 1 &\Rightarrow \Gamma^{\mathsf{NN}}_{xy} = 0 \end{aligned} \tag{9}$$

where $d(\cdot,\cdot)$ denotes hamming distance. Nearest neighbor adversary matrices were studied by Aaronson et al [ABDK+21], where they are shown to be closely related to the so-called spectral sensitivity, $\lambda(f)$. To define this quantity for a boolean function $f : \{0,1\}^N \to \{0,1\}$, let G_f be a subgraph of the boolean hypercube on $\{0,1\}^N$ with edges only between inputs distinguished by f. In particular, the vertex set of G_f is $\{0,1\}^N$, and there is an edge between strings x and y if and only if (a) $d(x,y) = 1$ and (b) $f(x) \neq f(y)$. Then let A_f be the adjacency matrix of G_f; we define $\lambda(f) = \|A_f\|$. The following theorem was established by Aaronson et al.

Theorem 60 ([ABDK+21], Theorem 10)**.** *For any total boolean function $f : \{0,1\}^N \to \{0,1\}$ and nearest neighbor adversary matrices Γ^{NN} for f, we have*

$$\max_{\Gamma^{NN}} \frac{\|\Gamma^{NN}\|}{\max_{i\in[N]} \|\Gamma^{NN}\|} = \lambda(f)$$

This quantity will play an important role in our parallel lower bound. A straightforward corollary of the above is that spectral sensitivity is upper bounded by quantum query complexity, $\lambda(f) = O(\mathsf{Q}(f))$ for all total f.

Furthermore, let $\mathcal{F}^{(f)}_{p-\text{res}}$ denote the set of all functions obtained from f by fixing all but p bits in the input. Formally, a function $g : \{0,1\}^p \to \{0,1\}$ is in $\mathcal{F}^{(f)}_{p-\text{res}}$ if there is a subset $S \subset [N]$ of size $|S| = p$ and an assignment $A : ([N] \setminus S) \to \{0,1\}$ of the remaining bits such that for all X satisfying $X_i = A(i)$ when $i \in ([N] \setminus S)$, we have $f(X) = g(X_S)$ where X_S is the concatenation of X_j for all $j \in S$.

Theorem 61. *For any total boolean function $f : \{0,1\}^N \to \{0,1\}$, the parallel quantum query complexity satisfies the following.*

$$\mathsf{Q}^{p\|}(f) = \Omega\left(\lambda(f) \cdot \frac{1}{\max_{g\in\mathcal{F}^{(f)}_{p-res}} \lambda(g)}\right)$$

We defer the proof of this theorem to Section 7.5. A slightly weaker but sometimes easier to reason about formulation of this lower bound is given by recalling that $\lambda(f) = O(\mathsf{Q}(f))$. This form, stated below, is the one we will apply to read-once formulas.

Corollary 62. *For any total boolean function $f : \{0,1\}^N \to \{0,1\}$, the parallel quantum query complexity satisfies the following.*

$$\mathsf{Q}^{p\|}(f) = \Omega\left(\lambda(f) \cdot \frac{1}{\max_{g\in\mathcal{F}^{(f)}_{p-res}} \mathsf{Q}(g)}\right)$$

The first term in this expression is a standard boolean complexity measure that is well understood for many functions. The second term is the inverse of the sequential query complexity of a worst-case restriction of f, and so the second term can be lower bounded by giving an algorithm for worst-case restrictions of f (upper bounding the denominator). This will allow us to prove parallel lower bounds with only sequential reasoning.

7.3 Read once formulas

Recall that a read-once formula is a boolean formula of variables $x_1, ..., x_N$ that can be written using and ($\wedge$), or ($\vee$), and not ($\neg$) in such a way that each variable appears exactly once. For example, the AND $\circ$ OR tree corresponding to

$$\psi_{\text{AND}\circ\text{OR}} = (x_1 \vee ... \vee x_{\sqrt{N}}) \wedge ... \wedge (x_{N-\sqrt{N}+1} \vee ... \vee x_N)$$

is a read once formula. It is well known that the AND $\circ$ OR tree has sequential quantum query complexity $\Omega(\sqrt{N})$ [BS04, Rei11a]. We give a lower bound for the parallel quantum query complexity of AND $\circ$ OR and more broadly any read-once formula, eliminating the possibility of large parallel query advantage in this case.

Theorem 63. *The p-parallel quantum query complexity of any read-once formula with N variables, $f : \{0,1\}^N \to \{0,1\}$ satisfies the following.*

$$\mathrm{Q}^{p\|}(f) = \Omega\left(\sqrt{\frac{N}{p}}\right)$$

Proof. We will apply Corollary 62, bounding the first and second term separately. A result of Barnum and Saks [BS04] provides a Γ for any read once formula f that satisfies $\| \Gamma \| = \Omega(\sqrt{N})$ and $\| \Gamma_i \| = 1$ for any $i \in [N]$ and is nearest neighbor. Hence we have $\lambda(f) = \Omega(\sqrt{N})$. To bound the second term we can simply observe that a p-restriction of a read-once formula is another read-once formula of size p. Consider a disjunction of the form

$$c = x_1 \vee x_2 \vee ... \vee x_k$$

If $x_i = 0$ is fixed then it can simply be removed from c, if $x_i = 1$ is fixed then c can be removed and replaced with a 1 literal (or vice versa if $\neg x_i$ appears). The rule for eliminating literals in a conjunction follows similarly. This procedure can be applied recursively until all fixed variables have been eliminated, leaving only p many free variables for a p-restriction. The quantum query complexity of a size p read-once formula is $O(\sqrt{p})$ [Rei11a], so the second term is $\Omega(1/\sqrt{p})$. □

7.4 Symmetric functions

We can also use Corollary 62 to reproduce parallel quantum lower bounds for symmetric functions. These bounds were implicit in the work of Grover and Radhakrishnan [GR04], but are arguably simpler to show once our theorem has been established. A symmetric function is a function $f : \{0,1\}^N \to \{0,1\}$ which satisfies

$$f(x) = f(\pi(x)) \qquad \text{for all permutations } \pi.$$

Or equivalently, $f(x)$ depends only on the hamming weight of x, $|x|$. We denote by $f_{|x|}$ the value of f on inputs of hamming weight $|x|$. We recall a standard combinatorial adversary lower bound for symmetric functions.

Let f be a non-constant symmetric total function from $\{0,1\}^N$ to $\{0,1\}$, and let $t_f \leq N/2$ be chosen such that

(1) $f_{t_f} \neq f_{t_f+1}$ or $f_{N-t_f} \neq f_{N-t_f-1}$

(2) t_f is chosen to minimize $|N/2 - t_f|$

WLOG let us assume that $f_{t_f} \neq f_{t_f+1}$ (if this is not the case then consider the complement function $\tilde{f}$ defined by $\tilde{f}(x) = f(\tilde{x})$, with $\tilde{x}$ being the bitwise complement). Consider the adversary matrix given by

$$\Gamma_{xy} = \begin{cases} 1 & \text{if } |x| = t_f, |y| = t_f + 1, |x \oplus y| = 1 \\ 0 & \text{otherwise} \end{cases}$$

This provides a tight $\Omega(\sqrt{Nt_f})$ sequential lower bound [BBC+98], and it is worth noting that this is also a nearest-neighbor adversary matrix. We are now ready to show the parallel lower bound.

Theorem 64. *For any non-constant symmetric function $f : \{0,1\}^N \to \{0,1\}$, define $t_f \leq N/2$ such that*

(1) $f_{t_f} \neq f_{t_f+1}$ or $f_{N-t_f} \neq f_{N-t_f-1}$

(2) t_f minimizes $|N/2 - t_f|$

Then the parallel quantum query complexity of f satisfies

$$Q^{p\|}(f) = \tilde{\Theta}\left(\sqrt{\frac{Nt_f}{p \min\{p, t_f\}}}\right)$$

Proof. We will again apply Corollary 62, bounding each term separately. From the preceding paragraph, we have

$$\lambda(f) = \Omega\left(\sqrt{Nt}\right)$$

bounding the first term. We will now upper bound $\max_{g \in \mathcal{F}^{(f)}_{p-\text{res}}} \mathsf{Q}(g)$ by giving an efficient quantum algorithm for a p-restriction of f. Let g be some p-restriction of f and $A : [N] \setminus S \to \{0,1\}$ be the assignment to the remaining $N - p$ variables. Let there be k many 1's in the assignment A. g is now a symmetric function "shifted" by k, i.e. we have

$$f_l = f_{l+1} (\Leftrightarrow) g_{l-k} = g_{l-k+1}$$

it follows that $t_g \leq t_f$, and we obviously have $t_g \leq p$ as the domain is size p. Hence there is a quantum algorithm for g with complexity $O(\sqrt{p \min\{p, t_f\}})$, and therefore

$$\frac{1}{\max_{g \in \mathcal{F}^{(f)}_{p-\text{res}}} \mathsf{Q}(g)} = \Omega\left(\frac{1}{\sqrt{p \min\{p, t_f\}}}\right)$$

$$\mathsf{Q}^{p\|}(f) = \Omega\left(\sqrt{\frac{Nt_f}{p \min\{p, t_f\}}}\right) \qquad \text{(Corollary 62)}$$

This is tight by a small modification of the algorithm presented by Grover and Radhakrishnan [GR04]. □

7.5 Proof of nearest neighbour lower bound

Now, we give a proof of the nearest neighbour lower bound stated in Theorem 61. By Theorem 60, the following formulation is equivalent:

Theorem (Reformulation of Theorem 61)**.** *For any total boolean function $f : \{0,1\}^N \to \{0,1\}$, and nearest neighbor adversary Γ^{NN} for f, the parallel quantum query complexity satisfies the following.*

$$\mathsf{Q}^{p\|}(f) = \Omega\left(\frac{\|\Gamma^{NN}\|}{\max_i \|\Gamma_i^{NN}\|} \cdot \frac{1}{\max_{g \in \mathcal{F}_{p-res}^{(f)}} \lambda(g)}\right)$$

Proof. Whenever we max over S in this proof, it is understood that $S \subseteq [N], |S| = p$, and when we max over i, it is understood that $i \in [N]$. From the parallel adversary method in Equation (1), we have

$$\mathsf{Q}^{p\|}(f) = \Omega\left(\frac{\| \Gamma^{\mathsf{NN}} \|}{\max_S \| \Gamma_S^{\mathsf{NN}} \|}\right) \tag{10}$$

By Lemma 65, we have that Γ_S^{NN} is a block diagonal matrix. Labelling each block by b such that $(\Gamma_S^{\mathsf{NN}})^{(b)}$ refers to the b^{th} block matrix, we get

$$\left\|\Gamma_S^{\mathsf{NN}}\right\| = \max_b \left\|\left(\Gamma_S^{\mathsf{NN}}\right)^{(b)}\right\| = \left\|\left(\Gamma_S^{\mathsf{NN}}\right)^{(b^*)}\right\| \tag{11}$$

where b^* is the block index which achieves the maximum in Equation (11).

By Lemma 65, we have that each of the block matrices $\left(\Gamma_S^{\mathsf{NN}}\right)_b$ are in fact adversary matrices for some function $g \in \mathcal{F}_{p-\text{res}}^{(f)}$. Therefore, for all b,

$$\frac{\left\|\left(\Gamma_S^{\mathsf{NN}}\right)^{(b)}\right\|}{\max_i \left\|\left(\Gamma_S^{\mathsf{NN}}\right)_i^{(b)}\right\|} \leq \max_{g \in \mathcal{F}_{p-\text{res}}^{(f)}} \mathsf{Q}(g) \tag{12}$$

Since Γ_S^{NN} is block diagonal, $\Gamma_S^{\mathsf{NN}}[x,y] = 0$ whenever $x_{S^{\complement}} \neq y_{S^{\complement}}$. Therefore, $\left(\Gamma_S^{\mathsf{NN}}\right)_i = \Gamma_{S \cap \{i\}}^{\mathsf{NN}}$. In particular, $\max_i \left\|\left(\Gamma_S^{\mathsf{NN}}\right)_i^{(b)}\right\| = \max_i \left\|\left(\Gamma_i^{\mathsf{NN}}\right)^{(b)}\right\|$. Putting this together with Equation (11) and Equation (12), we get

$$\left\|\Gamma_S^{\mathsf{NN}}\right\| \leq \left(\max_i \left\|\left(\Gamma_i^{\mathsf{NN}}\right)^{(b^*)}\right\|\right)\left(\max_{g \in \mathcal{F}_{p-\text{res}}^{(f)}} \mathsf{Q}(g)\right). \tag{13}$$

which when plugged into Equation (10) proves the theorem. □

Lemma 65. *Let Γ^{NN} be a Nearest Neighbour adversary matrix (Equation (9)) and let $S \subseteq [N]$ with $|S| = p$. Then,*

1. *Γ_S^{NN} is a block diagonal matrix with blocks of size $2^p \times 2^p$.*
2. *Each block of Γ_S^{NN} is an adversary matrix of a function $g \in \mathcal{F}_{p-res}^{(f)}$ where $\mathcal{F}_{p-res}^{(f)}$ is as defined in Section 7.2.*

Proof. Since we can relabel the inputs arbitrarily, we can assume, without loss of generality, that S contains the last p indices, that is $S = [N-p+1, N]$. Thus, we can view Γ_S^{NN} as a block matrix where each block, indexed by an assignment of $S^{\complement}$, is of size $2^p \times 2^p$. Recall that $\Gamma_S^{\mathsf{NN}}[x,y] \neq 0$ only if x and y differ at a single bit i with $i \in S$. This means that for any x, y with $x_{S^{\complement}} \neq y_{S^{\complement}}$, we have $\Gamma_S^{\mathsf{NN}}[x,y] = 0$. This completes the proof of part 1.

For any assignment b of $S^{\complement}$, we define the induced function $f_b : \{0,1\}^p \to \{0,1\}$ as

$$f_b(z) \coloneqq f(z \frown b)$$

where $z \frown b$ corresponds to an input x with $x_S = z$ and $x_{S^{\complement}} = b$.

Clearly, $f_b \in \mathcal{F}^{(f)}_{p-\mathrm{res}}$. Let $\left(\Gamma^{\mathsf{NN}}_S\right)^{(b)}$ be the diagonal block of $\left(\Gamma^{\mathsf{NN}}_S\right)$ associated with the assignment b. It remains to verify that $\left(\Gamma^{\mathsf{NN}}_S\right)^{(b)}$ is an adversary matrix for f_b. We know that $\left(\Gamma^{\mathsf{NN}}_S\right)^{(b)}$ is symmetric since Γ^{NN}_S is symmetric. Moreover, for any $z, z' \in \{0,1\}^p$, we have

$$f_b(z) = f_b(z') \implies f(z \frown b) = f(z' \frown b) \implies (\Gamma^{\mathsf{NN}}_S)_{z\frown b, z'\frown b} = 0 \implies \left((\Gamma^{\mathsf{NN}}_S)^{(b)}\right)_{z,z'} = 0.$$

This proves part 2. □

Acknowledgments

The authors thank Laxman Dhulipala for suggesting to investigate the relationship between quantum algorithms and parallelism, Luke Schaeffer and Chaitanya Karamchedu for helpful discussions, and Andrew Childs for valuable feedback on an earlier draft. We also thank an anonymous reviewer for largely simplifying our constructions in Section 5. ASG additionally thank Stacey Jeffery and Ronald de Wolf for helpful discussions.

JC is supported by the US Department of Energy grant no. DESC0020264. ASG is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing and Quantum Testbed Pathfinder programs (award numbers DE-SC0020312 and DE-SC0019040). MV is supported by the US Department of Energy grant no. DESC0020264 as well as the US Department of Energy Nuclear Energy University Programs award number DE-NE0009417.

References

[AA15] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, page 307–316, 2015. doi:10.1145/2746539.2746547.

[Aar10] Scott Aaronson. Bqp and the polynomial hierarchy. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, page 141–150, 2010. doi:10.1145/1806689.1806711.

[ABB+17] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *Journal of the ACM*, 64(5), 2017. doi:10.1145/3106234.

[ABBD+16] Anurag Anshu, Aleksandrs Belovs, Shalev Ben-David, Mika Göös, Rahul Jain, Robin Kothari, Troy Lee, and Miklos Santha. Separations in communication complexity using cheat sheets and information complexity. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 555–564, 2016. doi:10.1109/FOCS.2016.66.

[ABDK+21] Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, page 1330–1342, 2021. doi:10.1145/3406325.3451047.

[ABK16] Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, page 863–876. Association for Computing Machinery, 2016. doi:10.1145/2897518.2897644.

[ACC+23] Atul Singh Arora, Andrea Coladangelo, Matthew Coudron, Alexandru Gheorghiu, Uttam Singh, and Hendrik Waldner. Quantum depth in the random oracle model. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1111–1124, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585153.

[Amb02] Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. doi:10.1006/jcss.2002.1826.

[BACS07] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007. doi:10.1007/s00220-006-0150-x.

[BBC+98] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, 1998. doi:10.1145/502090.502097.

[Bd02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. Complexity and Logic. doi:10.1016/S0304-3975(01)00144-X.

[BDB20] Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions. In *2020 IEEE 61st Annual*

Symposium on Foundations of Computer Science (FOCS), pages 240–246, 2020. doi:10.1109/FOCS46700.2020.00031.

[BGK18] Sergey Bravyi, David Gosset, and Robert Koenig. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018. URL: https://www.science.org/doi/abs/10.1126/science.aar3106, arXiv:https://www.science.org/doi/pdf/10.1126/science.aar3106, doi:10.1126/science.aar3106.

[BLZ21] Jeremiah Blocki, Seunghoon Lee, and Samson Zhou. On the Security of Proofs of Sequential Work in a Post-Quantum World. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*, volume 199 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:27, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITC.2021.22, doi:10.4230/LIPIcs.ITC.2021.22.

[BS04] Howard Barnum and Michael Saks. A lower bound on the quantum query complexity of read-once functions. *Journal of Computer and System Sciences*, 69(2):244–258, 2004. doi:10.1016/j.jcss.2004.02.002.

[BS13] Aleksandrs Belovs and Robert Spalek. Adversary lower bound for the k-sum problem. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, page 323–328, 2013.

[BS21] Nikhil Bansal and Makrand Sinha. K-forrelation optimally separates quantum and classical query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, page 1303–1316. Association for Computing Machinery, 2021. doi:10.1145/3406325.3451040.

[Bur19] Paul Burchard. Lower bounds for parallel quantum counting. 2019. arXiv:1910.04555.

[CCD+03] Andrew Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, page 59–68, 2003. doi:10.1145/780542.780552.

[CCL23] Nai-Hui Chia, Kai-Min Chung, and Ching-Yi Lai. On the need for large quantum depth. *J. ACM*, 70(1), jan 2023. doi:10.1145/3570637.

[CFHL21] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 598–629, Cham, 2021. Springer International Publishing.

[CH22] Nai-Hui Chia and Shih-Han Hung. Classical verification of quantum depth, 2022. arXiv:2205.04656.

[CH23] Nai-Hui Chia and Shih-Han Hung. Non-interactive classical verification of quantum depth: A fine-grained characterization. Cryptology ePrint Archive, Paper 2023/1911, 2023. https://eprint.iacr.org/2023/1911. URL: https://eprint.iacr.org/2023/1911.

[CKM+23] Sourav Chakraborty, Chandrima Kayal, Rajat Mittal, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh. On the Composition of Randomized Query Complexity

and Approximate Degree. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275, pages 63:1–63:23, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.63.

[CM20] Matthew Coudron and Sanketh Menda. Computations with greater quantum depth are strictly more powerful (relative to an oracle). In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 889–901, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384269.

[CP18] Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 451–467, Cham, 2018. Springer International Publishing.

[CW00] R. Cleve and J. Watrous. Fast parallel circuits for the quantum fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536, 2000. doi:10.1109/SFCS.2000.892140.

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992. doi:10.1098/rspa.1992.0167.

[GHMP02] Frederic Green, Steven Homer, Cristopher Moore, and Christopher Pollett. Counting, fanout and the complexity of quantum acc. 2(1):35–65, dec 2002.

[GR04] Lov K. Grover and Jaikumar Radhakrishnan. Quantum search for multiple items using parallel queries. 2004. URL: https://api.semanticscholar.org/CorpusID:16025437.

[Gro96] Lov Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, page 212–219, 1996. doi:10.1145/237814.237866.

[GSTW23] Uma Girish, Makrand Sinha, Avishay Tal, and Kewen Wu. The power of adaptivity in quantum query algorithms, 2023. arXiv:2311.16057.

[HLS07] Peter Hoyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 526–535, 2007.

[HŠ05] Peter Høyer and Robert Špalek. Quantum fan-out is powerful. *Theory of Computing*, 1(5):81–103, 2005. URL: https://theoryofcomputing.org/articles/v001a005, doi:10.4086/toc.2005.v001a005.

[JMW17] Stacey Jeffery, Frederic Magniez, and Ronald Wolf. Optimal parallel quantum query algorithms. *Algorithmica*, 79(2):509–529, 2017. doi:10.1007/s00453-016-0206-z.

[Kim12] Shelby Kimmel. Quantum adversary (upper) bound. In *Automata, Languages, and Programming*, pages 557–568, 2012.

[KNTSZ01] Hartmut Klauck, Ashwin Nayak, Amnon Ta-Shma, and David Zuckerman. Interaction in quantum communication and the complexity of set disjointness. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, page 124–133, 2001. doi:10.1145/380752.380786.

[Mon10] Ashley Montanaro. Nonadaptive quantum query complexity. *Information Processing Letters*, 110(24):1110–1113, 2010.

[Mon13] Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 20, 2013. doi:10.4086/cjtcs.2014.006.

[PRS01] Stephen Ponzio, Jaikumar Radhakrishnan, and Venkatesh Srinivasan. The communication complexity of pointer chasing. *J. Comput. Syst. Sci.*, 62:323–355, 03 2001. doi:10.1006/jcss.2000.1731.

[Rei11a] Ben W. Reichardt. Faster quantum algorithm for evaluating game trees. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, page 546–559, 2011.

[Rei11b] Ben W. Reichardt. *Reflections for quantum query algorithms*, pages 560–569. 2011. doi:10.1137/1.9781611973082.44.

[Sho94] Peter Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi:10.1109/SFCS.1994.365700.

[Sim97] Daniel Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. doi:10.1137/S0097539796298637.

[SSW21] Alexander A. Sherstov, Andrey A. Storozhenko, and Pei Wu. An optimal separation of randomized and quantum query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, page 1289–1302, 2021. doi:10.1145/3406325.3451019.

[Tal13] Avishay Tal. Properties and applications of boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, page 441–454, 2013. doi:10.1145/2422436.2422485.

[TT16] Yasuhiro Takahashi and Seiichiro Tani. Collapse of the hierarchy of constant-depth exact quantum circuits. *computational complexity*, 25(4):849–881, 2016. doi:10.1007/s00037-016-0140-0.

[WKST19] Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 515–526, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316404.

[WP23] Adam Bene Watts and Natalie Parham. Unconditional Quantum Advantage for Sampling with Shallow Circuits. 1 2023. arXiv:2301.00995.

[Zal99] Christof Zalka. Grover's quantum searching algorithm is optimal. *Phys. Rev. A*, 60:2746–2751, Oct 1999. URL: https://link.aps.org/doi/10.1103/PhysRevA.60.2746, doi:10.1103/PhysRevA.60.2746.

[Zha04] Shengyu Zhang. On the power of ambainis's lower bounds. In *Automata, Languages and Programming*, pages 1238–1250, 2004.

A Composition theorems for parallel query complexity

In this section, we will present composition theorems for parallel quantum query complexity that will be useful for us.

A.1 A composition theorem for deterministic parallel query complexity

In this section, we prove a deterministic parallel composition theorem that is useful for showing deterministic parallel query lower bounds in Section 5. In particular, we show a tight (up to constant factors) composition theorem for the case when the deterministic query complexity of the inner function is close to being full. The same technique would apply for the case when the outer function has $\Theta(n)$ deterministic query complexity. We leave (dis)proving a general deterministic parallel composition theorem as an open problem.

Theorem 66. *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be (partial) functions, where $\mathcal{D}_f \subseteq \{0,1\}^N, \mathcal{D}_g \subseteq \{0,1\}^M$ such that $\mathsf{D}(g) = \Theta(M)$ and $p \in [MN]$. Then, $\mathsf{D}^{p\|}(f \circ g) = \Theta\left(\left\lceil \frac{M}{p} \right\rceil \mathsf{D}^{\lceil p/M \rceil \|}(f)\right)$.*

Proof. We first describe a deterministic p-parallel algorithm. Let $\mathcal{A}_f$ be an optimal deterministic $\lceil p/M \rceil$-parallel query algorithm for f. We will make queries in rounds where in each round, we will choose $\lceil p/M \rceil$ inputs for f that the algorithm $\mathcal{A}$ would have chosen to compute and make $\lceil M/p \rceil p$ non-adaptive queries to completely determine the inputs of g corresponding to these $\lceil p/M \rceil$ inputs. Thus, each round takes $\lceil M/p \rceil$ p-parallel queries to compute these $\lceil p/M \rceil$ inputs to f. Due to the optimality of $\mathcal{A}$, the total number of rounds needed to compute $f \circ g$ is $\mathsf{D}^{\lceil p/M \rceil}(f)$ and, therefore, $\lceil M/p \rceil \mathsf{D}^{\lceil p/M \rceil}(f)$ queries suffices.

For the lower bound, we will use the adversarial argument against a deterministic p-parallel algorithm $\mathcal{A}$ computing $f \circ g$. We consider two cases: i) $p \leq M$ and, ii) $p > M$. The first case is trivial since any p-parallel deterministic query algorithm must make $\Omega\left(\frac{\mathsf{D}(f)\cdot\mathsf{D}(g)}{p}\right) = \Omega\left(\frac{M}{p}\mathsf{D}(f)\right)$ p-parallel queries to compute $f \circ g$ against an adversarial oracle. For the second case, our oracle will allow $\mathcal{A}$ to make $2p$-parallel queries as long as for each such query, $\mathcal{A}$ uses p of these non-adaptive queries to find all the inputs to p/M of the g functions of its choice and, thus, determining p/M inputs to f. This oracle is at least as powerful as a regular p-parallel query oracle since it allows computing p/M inputs to f along with making p additional non-adaptive queries to the inputs to g. Thus, it suffices to lower bound the number of queries to this oracle. Since $\mathcal{A}$ is required to (non-adaptively) compute p/M inputs to f for each $2p$-parallel query that it makes, the minimal number of total inputs to f that $\mathcal{A}$ must compute is $p/M \cdot \mathsf{D}^{p/M\|}$. This means that $\mathcal{A}$ must query at least $p/M \cdot \mathsf{D}(g)\mathsf{D}^{p/M\|} = \Omega(p\mathsf{D}^{p/M\|}$ total bits of input of $f \circ g$. It follows that $\mathcal{A}$ would need $\Omega(\mathsf{D}^{p/M\|}$ $2p$-parallel queries to compute $f \circ g$. □

We state the implication of this theorem that we use in Section 5.

Corollary 67. *Let $f : \mathcal{D}_f \to \{0,1\}$ be a (partial) function, where $\mathcal{D}_f \subseteq \{0,1\}^N$ and $p \in [N^3]$. Then, $\mathsf{D}^{p\|}(f \circ (AND \circ OR)_{N^2}) = \Theta\left(\left\lceil \frac{N^2}{p} \right\rceil \mathsf{D}^{\lceil p/N^2 \rceil}(f)\right)$.*

We believe that the above theorem could be generalized to prove the following conjecture.

Conjecture 68. *Let $f : \mathcal{D}_f \to \{0,1\}$ and $g : \mathcal{D}_g \to \{0,1\}$ be (partial) functions, where $\mathcal{D}_f \subseteq \{0,1\}^N, \mathcal{D}_g \subseteq \{0,1\}^M$ and $p \in [MN]$. Then, $\mathsf{D}^{p\|}(f \circ g) = \Theta\left(\min_{q\in[M]} \mathsf{D}^{q\|}(g)\mathsf{D}^{\lceil p/q \rceil \|}(f)\right)$.*

A.2 A composition theorem for randomized parallel query complexity

Below, we provide a randomized parallel composition theorem that will help us establish our separations in Section 5. This result is analogous to and inspired by Theorem 5 of [ABK16], which shows the randomized composition theorem for sequential query complexity with AND or OR (or AND$\circ$OR) being the inner function. Even though we restrict the inner function to be k-SUM or BLOCK k-SUM (or BKK), our proof technique works for any problem that the AND or OR problem could be directly reduced to. Intuitively, the idea is to construct an algorithm for computing f using an algorithm that computes $f \circ$ BKK with lesser parallelism but not too many more queries by choosing the instance of BKK carefully.

Theorem 69. *Let $f : \mathcal{D} \to \{0,1\}$ be a (partial) function where $\mathcal{D} \subseteq \{0,1\}^N$, and let k-SUM$_M$, BLOCK k-SUM$_M$ and BKK$_{M^2}$ be defined as in Problem 9, Problem 10 and Problem 11, taking $k = \log N$. Then, $R^{p\|}(f \circ k\text{-SUM}_M) = \tilde{\Omega}\left(\left\lceil \frac{M}{p} \right\rceil R^{\lceil p/M \rceil \|}(f)\right)$ and $R^{p\|}(f \circ \text{BLOCK } k\text{-SUM}_M) = \tilde{\Omega}\left(\left\lceil \frac{M}{p} \right\rceil R^{\lceil p/M \rceil \|}(f)\right)$. Therefore, $R^{p\|}(f \circ \text{BKK}_{M^2}) = \tilde{\Omega}\left(\left\lceil \frac{M^2}{p} \right\rceil R^{\lceil p/M^2 \rceil \|}(f)\right)$.*

Proof. We give the proof for k-SUM$_M$ here; the proof for BLOCK k-SUM$_M$ proceeds exactly in the same way. Consider a randomized algorithm $\mathcal{A}$ that computes $f \circ k$-SUM$_M$ on string Y in t many p-parallel queries to Y, where $|Y| = MN$. Then we claim that there exists a randomized algorithm $\mathcal{B}$ which can compute f on any string X using $\tilde{O}(tp/M)$ total queries to X, where $|X| = N$. To show this claim, consider an algorithm $\mathcal{A}$ that can compute $f \circ k$-SUM in t many p-parallel queries.

On input $X := x_1x_2...x_N$, $\mathcal{B}$ proceeds as follows:

1. Construct an input Y, a bit string of length MN in the following way. Divide Y into N blocks Y_i each of size M. Further divide each Y_i into sub blocks of size $10k \log M$ each and interpret each sub block as a number in the range 1 to $\Omega(M^k)$. We represent the j^{th} sub block of Y_i as Y_{ij}. For each i from 1 to N, do the following:
   (a) For each $j \in \{1, 2, ..., k-1\}$, set Y_{ij} as 0.
   (b) Select z from $\left\{k, k+1, ..., \frac{M}{10k \log M}\right\}$ uniformly at random. Set Y_{iz} as $\overline{x_i}$
   (c) For all $j \notin \{1, 2, ..., k-1\}$ and $j \neq z$, set Y_{ij} as 1
2. Run algorithm $\mathcal{A}$ on Y and return the output.

Notice that k-SUM$_M(Y_i) = x_i$ which implies $f \circ k$-SUM$_M(Y) = f(X)$. For every input $\mathcal{B}$ prepares, $\mathcal{A}$ correctly computes $f \circ k$-SUM$_M(Y)$ with probability $\geq 2/3$, hence $\mathcal{B}$ also correctly computes $f(X)$ with probability $2/3$ where the probability is taken over both $\mathcal{A}$'s randomness as well as the randomness in preparing the inputs.

Since algorithm $\mathcal{A}$ makes t many p-parallel queries to Y, the total number of sub blocks Y_{ij} queried is at most tp. Let Y' be the string made from Y ignoring the first $k-1$ sub blocks in every block. Now, we invoke Lemma 70 for string Y' setting the variables $n = N$, $m = \frac{M}{10k \log M} - (k-1)$, $l = tp$ and the $*$'s in the position of the x_i's. It follows that with probability $\geq 9/10$ over the inputs that $\mathcal{B}$ prepares, $\tilde{O}(tp/M)$ total queries are made to the x_i's. Since for every input $\mathcal{B}$ prepares, $\mathcal{A}$ computes $f \circ k$-SUM$_M$ correctly with probability $2/3$, the probability that $\mathcal{B}$ computes f correctly, while also making $\tilde{O}(tp/M)$ total queries to X is $\frac{2}{3} \times \frac{9}{10} = \frac{3}{5}$. This is constant away from $1/2$ and this proves the claim.

It remains to show that this algorithm can be simulated in low depth with the requisite amount of parallelism. We have two cases, (1) $p \leq M$ and (2) $p > M$.

Case 1 When $p \leq M$, the statement becomes $\mathsf{R}^{p\|}(f \circ k\text{-SUM}_M) = \tilde{\Omega}\left(\frac{M\mathsf{R}(f)}{p}\right)$. We know algorithm $\mathcal{B}$ makes $\tilde{O}(tp/M)$ total queries to X to compute f, so we can just do them all sequentially. Thus,

$$\frac{tp}{M} = \tilde{\Omega}\left(\mathsf{R}(f)\right) \implies t = \tilde{\Omega}\left(\frac{M\mathsf{R}(f)}{p}\right)$$

which implies the statement.

Case 2 When $p > M$, the statement becomes $\mathsf{R}^{p\|}(f \circ k\text{-SUM}_M) = \tilde{\Omega}\left(\mathsf{R}^{p/M\|}(f)\right)$. Here, we simulate algorithm $\mathcal{B}$ using t many p/M-parallel queries. Let $r_1, r_2, ..., r_t$ be the number of queries in every layer of $\mathcal{B}$ made to an element of X. Observe that to simulate layer i using p/M parallelism, we require $\left\lceil \frac{r_i}{p/M} \right\rceil$ many p/M-parallel queries. Thus, the total number of p/M-parallel queries we require is:

$$\sum_{i=1}^{t} \left\lceil \frac{r_i}{p/M} \right\rceil \leq \sum_{i=1}^{t} \left(1 + \frac{r_i}{p/M}\right) \leq t + \frac{\sum_i r_i}{p/M} \leq t + \frac{\tilde{O}(tp/M)}{p/M} = \tilde{O}(t).$$

Thus, we must have $t = \tilde{\Omega}\left(\mathsf{R}^{p/M\|}(f)\right)$ which implies the statement.

□

Below is a helper lemma we require to prove Theorem 69.

Lemma 70. *Let $F_i \in [M]^m$ be any fixed string for $i \in [n]$, and construct $Y_i \in ([M] \cup \{*\})^m$ by replacing a random character in F_i with a $*$. Further define $Y = Y_1 \parallel ... \parallel Y_n$. Any randomized algorithm (even ones that depend on F_i) that makes l queries to Y will query at most $20l/m$ many $*$'s with probability at least $9/10$.*

Proof. Let $\mathcal{A}$ be some randomized algorithm making l queries to Y, and consider a randomly chosen Y_i. WLOG we can assume that $\mathcal{A}$ does not query a Y_i after it has queried on a $*$— any algorithm that continued to query past this point could be given answers from F_i without more queries to Y_i. If $\mathcal{A}$ makes l queries total then in expectation it makes l/n queries to Y_i.

We can write the strategy of $\mathcal{A}$ on Y_i as a distribution $\mathcal{D}_T$ over decision trees (where the randomness now ranges over the randomness of $\mathcal{A}$ as well as the $*$ placement in other $Y_{j\neq i}$). Further, every node v in decision tree T from $\mathcal{D}_T$ has only two possibilities: the queried value is a $*$, or it is from F_i. $\mathcal{A}$ stops querying after finding a $*$, so the tree is a caterpillar tree of height h, and over the randomness of placing the $*$ the expected queries will be $(h+1)/2$. We therefore have

$$\begin{aligned}\mathbb{E}_{T\sim\mathcal{D}_T,Y}[\text{height}(T)] &= 2\mathbb{E}_{T\sim\mathcal{D}_T,Y}[\text{queries}_T(Y_i)] - 1 \\ &\leq \frac{2l}{n}\end{aligned}$$

A fixed tree T of height h finds a $*$ with probability at most h/m over random inputs, as the problem is an unstructured search problem. Hence the probability of finding the star in block Y_i is at most

$$\begin{aligned}\Pr_{T\sim\mathcal{D}_T,Y}[\text{star found in } Y_i] &\leq \frac{\mathbb{E}_{T\sim\mathcal{D}_T,Y}[\text{height}(T)]}{m} \\ &\leq \frac{2l}{nm}\end{aligned}$$

By linearity of expectation, the expected total number of 1's found over all blocks is at most $2l/m$. A straightforward application of Markov's inequality now proves the claim. □

B Parallel query complexity in the cheatsheet framework

In this section, we present lower bounds for query complexity measures in the cheatsheet framework.

B.1 Deterministic parallel lower bound

In this section, we will show that having access to a cheatsheet for a function f is not too helpful for a p-parallel deterministic algorithm for computing f. Intuitively, the size of the cheatsheet is big enough that searching through it will already be too expensive for any such algorithm. Our result is a straightforward adaptation of Lemma 21 of [ABK16] but we repeat it here for completeness.

Theorem 71. *Let $f : \mathcal{D} \to \{0,1\}$ be a partial function, where $\mathcal{D} \subset [M]^N$, and let f_{cs} be the cheat sheet version of f with $c = 10 \log N$ independent copies of f. Then $\mathsf{D}^{p\|}(f_{\mathsf{cs}}) \geq \mathsf{D}^{p\|}(f)$.*

Proof. We will use the adversarial argument against a p-parallel deterministic algorithm $\mathcal{A}$ making computing f_{cs}. Suppose that $\mathcal{A}$ makes fewer than $\mathsf{D}^{p\|}(f)$ queries. For all the indices of a p-parallel query made by $\mathcal{A}$ that belongs to an input to one of the c copies of f, we will answer them as an adversary against a p-parallel deterministic algorithm for f would, and for all the indices of a p-parallel query made by $\mathcal{A}$ that belongs to a cheatsheet, we will answer a 0. Since $pD^{p\|}(f) \leq N$ and the number of cheatsheets are $2^c = \Omega(N^{10})$, $\mathcal{A}$ must not have queried a single bit in most of the cheatsheets. Furthermore, since $\mathcal{A}$ made less than $\mathcal{A}$ $\mathsf{D}^{p\|}(f)$ p-parallel queries in total, it must not have made at least $\mathsf{D}^{p\|}(f)$ p-parallel queries to any input to any f. Thus, it must not know the output to any f. Therefore, the partially revealed instance for f_{cs} could be completed to form a 0-instance and also completed to form a 1-instance. It follows that $\mathcal{A}$ must make at least $\mathsf{D}^{p\|}(f)$ queries. □

B.2 Cheatsheet randomized parallel lower bound

Similar to the previous section, we prove in this section that having access to a cheatsheet for a function f is not too helpful for a p-parallel randomized algorithm for computing f.

Our proof of this result (Theorem 73) is a straightforward adaptation of Lemma 6 of [ABK16] but we repeat it here for completeness. We begin with the following observation.

Observation 72. Let $f : \mathcal{D} \to \{0,1\}$ be a partial function, where $\mathcal{D} \subset [M]^n$, and let f_{cs} be the cheat sheet version of f with $c = 10 \log N$ copies of f. Let $\mathcal{A}$ be a p-parallel bounded-error randomized algorithm for f_{cs}. Let $x^1, x^2, \cdots, x^c \in \mathcal{D}$ and z be an input to f_{cs} comprised of inputs $x^1, x^2, \cdots, x^c \in \mathcal{D}$ to f with an all-zero cheat sheet[¶¶]. Let ℓ index the cheat sheet associated with the string $f(x^1)$, $f(x^2), \cdots f(x^c)$. Then, $\mathcal{A}$ must query at least 1 bit in the ℓ^{th} cheat sheet with probability at least $1/3$.

Proof. Let y be an input to f_{cs} comprised of inputs $x_1, x^2, \cdots, x^c \in \mathcal{D}$ to f and the same cheat sheet as z except for the ℓ^{th} cheat sheet being valid. That is, $f_{\mathsf{cs}}(y) = 1$. Since $\mathcal{A}$ computes f_{cs} with probability at least $2/3$, $f_{\mathsf{cs}}(y) \neq f_{\mathsf{cs}}(z)$ and y and z only differ at the ℓ^{th} cheat sheet, our desired claim follows. □

[¶¶]For any input z to f_{cs}, we assume, without loss of generality, that if the cheat sheet corresponding to $f^{\times c}(z)$ is all-zero, then $f_{\mathsf{cs}}(z) = 0$. As argued in [ABK16], this can be ensured by requiring all valid cheat sheets to begin with a 1.

Theorem 73. *Let $f : \mathcal{D} \to \{0,1\}$ be a partial function, where $\mathcal{D} \subset [M]^n$, and let f_{cs} be the cheat sheet version of f with $c = 10 \log N$ copies of f. Then $\mathsf{R}^{p\|}(f_{\mathsf{cs}}) = \Omega(\mathsf{R}^{p\|}(f)/c^2) = \tilde{\Omega}(\mathsf{R}^{p\|}(f))$.*

Proof. Let $\mathcal{D}^0$ and $\mathcal{D}^1$ be 0-input and 1-input distributions for f that is hard to distinguish for a p-parallel bounded-error randomized algorithm. In particular, the p-parallel randomized query complexity of distinguishing them with probability at least $1/2 + 1/12c$ is $\Omega(R^{p\|}(f)/c^2)$. Let $\mathcal{A}$ be any p-parallel bounded-error randomized algorithm for f_{cs}.

For each input z to f_{cs} with an all-zero cheat sheet and for each index $i \in [2^c]$ of a cheat sheet, let p_i^z be the probability that $\mathcal{A}$ queries the i^{th} cheat sheet on input z. Let ℓ_z index the cheat sheet associated with the string $f^{\times c}(z)$. From Observation 72, we know that $p_{\ell_z}^z \geq 1/3$. Furthermore, $\sum_{i\in[2^c]} p_i^z \leq p\mathsf{R}^{p\|}(f)$.

For each $j \in [2^c]$, let $\mathcal{D}^j = \mathcal{D}^{j_1} \times \mathcal{D}^{j_2} \times \cdots \times \mathcal{D}^{j_c}$. Let $q_i^j = \mathbb{E}_{z\sim\mathcal{D}^j} p_i^z$ for all $i, j \in [2^c]$. By the arguments made above, we can infer that $q_j^j \geq 1/3$ and $\sum_{i\in[2^c]} q_i^j \leq pR^{p\|}(f)$. This means that for all but $2^{c/2} = N^5$ many $i \in [2^c]$, $q_i^j \leq 1/(p\mathsf{R}^{p\|}(f))^5$. In particular, for each $j \in [2^c]$, there is an $i \in [2^c]$ such that $q_i^j \leq 1/6$.

Let $k \in [2^c]$ be an index such that $q_k^{0^c} \leq 1/6$. We argued above that $q_k^k \geq 1/3$. Let $0^c = k_0, k_1, \cdots, k_c = k$ be a sequence of indices such that $|k_{i-1} \oplus k_i| \leq 1$ for all $i \in [c]$. Therefore, there must exist an $i \in [c]$ with $q_k^{k_i} - q_k^{k_{i-1}} \geq 1/6c$ and $q_k^{k_{i-1}} \leq 1/3$. Without loss of generality, we can assume that $|k_i| - |k_{i-1}| = 1$.

Using $\mathcal{A}$, we now give an algorithm to distinguish $\mathcal{D}^0$ and $\mathcal{D}^1$ with probability at least $1/2 + 1/12c$. Given an input from $\mathcal{D}_0$ or $\mathcal{D}_1$, our algorithm samples an input from $\mathcal{D}^{k_{i-1}}$ or $\mathcal{D}^{k_i}$; this can be done since $|k_{i-1} \oplus k_i| \leq 1$. Therefore, deciding if the sampled input is from $\mathcal{D}^{k_{i-1}}$ or from $\mathcal{D}^{k_i}$ lets us decide whether the given input is from $\mathcal{D}_0$ or from $\mathcal{D}_1$.

We run the algorithm $\mathcal{A}$ on the sampled input with an all-zero cheat sheet, and if it queries a bit in the k^{th} cheat sheet, we output 1 and otherwise, we output 1 with probability $p = \max\{0, (1-a-b)/(2-a-b)\}$ where $a = q_k^{k_i}$ and $b = q_k^{k_{i-1}}$. If $p = 0$, then since $b \leq 1/3$, we have $a \geq 2/3$ so our algorithm succeeds with probability at least $2/3$. Otherwise, our algorithm succeeds with probability at least $1/2 + (a-b)/2 \geq 1/2 + 1/12c$: if the given input is sampled from $\mathcal{D}^1$, then the success probability is $a + (1-a)\frac{1-a-b}{2-a-b} \geq (1+a-b)/2$; otherwise, it is $(1-b)\frac{1}{2-a-b} \geq (1+a-b)/2$. But this means that our algorithm must make $\Omega(\mathsf{R}^{p\|}(f)/c^2)$ p-parallel randomized queries. Hence, $\mathcal{A}$ needs to make $\Omega(\mathsf{R}^{p\|}(f)/c^2)$ p-parallel randomized queries, which implies the desired result. □

C Pointer Chasing bounds

In this section, we present a proof for each of the parts of Theorem 16, beginning with a deterministic algorithm.

Deterministic parallel upper bound

Lemma 74. $\mathsf{D}^{p\|}(\textsc{Pointer Chasing}) \leq \min(k, N/p)$.

Proof. We will consider two cases.

1. $k \leq N/p$. For each $i \in [k]$, we can iteratively compute X^i using k sequential queries to the oracle for the input X. Thus, we will know the last bit of $X^k(0^n)$.

2. $k > N/p$. Using N/p p-parallel queries, we can compute the whole input X (and in particular, can compute the last bit of $X^k(0^n)$).

□

Deterministic parallel lower bound

Lemma 75. $\mathsf{D}^{p\|}(\textsc{Pointer Chasing}) = \Omega(\min(k, N/p))$.

Proof. Let $\mathcal{A}$ be a deterministic algorithm for pointer chasing. We will construct two instances f, g of pointer chasing such that f is a YES instance and g is a NO instance, but $\mathcal{A}$ does not distinguish f and g supposing that it makes fewer than $\min(k/10, N/10p)$ queries.

We will iteratively construct both instances, maintaining the invariant that the algorithm cannot "jump ahead" in the chain. In either case, the chain will be a path with no cycles. To answer a round of p-parallel queries, suppose that $\mathcal{A}$ has queried input function h at index/value pairs $(0, h(0)), (h(0), h(h(0))), ..., (h^{l-1}(0), h^l(0))$, as well as some other set S of index/value pairs that is of size smaller than $\min(k/10, N/10p)$. Suppose by induction that S contains only pairs of the form $(x, 0)$ (this is clearly true for an algorithm making no queries, and we will maintain it here). Let B denote the set of positions which are being queried in this round. We will then pick the value $y = h^l(0)$ such that y is not any of the indices in B, nor in S, nor does it create a chain in the path (this is always possible so long as fewer than $\min(k/10, N/10p) < N/2$ queries have been made). We thus maintain the invariant that after q rounds of queries, a chain of length at most q has been revealed. Therefore, if $\mathcal{A}$ makes $k-1$ rounds of queries (and supposing that it makes fewer than $N/10p$ total p-parallel queries), we can construct f in the manner outlined aboved, and finally choose $f^k(0)$ so the last bit is 1. Similarly for g, but choosing $g^k(0)$ such that the last bit is 0. Any input/value pairs not fixed by this procedure we can simply set the value to 0. □

Randomized parallel lower bound

Lemma 76. $\mathsf{R}^{p\|}(\textsc{Pointer Chasing}) = \Omega(\min(k, N/pk))$.

Proof. From Yao's lemma, it suffices to consider deterministic algorithms on some distribution of inputs. We will consider inputs that are random permutations, i.e. let $f : \{0,1\}^n \to \{0,1\}^n$ be a random bijective function. To reason about a deterministic algorithm we will consider it's query transcript, i.e. a list $(x, f(x))$ of all points that it has queried. After q rounds of p-parallel queries, this transcript is of size pq.

It suffices to consider $k-1$-round p-parallel algorithms; we will show that such algorithms must make at least $\Omega(N/k)$ total queries to succeed with constant probability. Suppose that the chain $(0, f(0)), ..., (f^{l-1}(0), f^l(0))$ of length l appears in the transcript, and further that no $f^j(0)$ for $l < j \leq k$ appears (e.g. the algorithm has learned the first l elements of the chain, but no more). Because $f^k(0)$ determines the final output, clearly such a deterministic algorithm cannot decide f with probability exceeding $1/2$. Note that this invariant form of the transcript trivially holds with $l = 0$ for an algorithm making no queries. In a round of p-parallel queries, we will bound the probability that the algorithm transitions from a transcript satisfying the invariant, to one that does not.

A round of p-parallel queries selects p positions to query, out of N total. The remaining $k-l$ elements in the chain do not occur in the list of q many queried elements, and so we can consider placing them randomly among the remaining $N-q$, which satisfies $N-q \geq N/2$ by assumption. We therefore have probability $2p(k-l)/N \leq 2pk/N$ probability that the remaining elements in this chain do

not intersect with the query set. If there is no intersection, then the chain length will increase by exactly 1 in this round of queries. Hence, in $k-1$ rounds we will break the invariant with probability $(k-1) * 2pk/N = O(pk^2/N)$ by the union bound. For this to be a constant we need $p = \Omega(N/k^2)$, which gives $\Omega(N/k)$ total queries. □

Quantum lower bound

Lemma 77. $\mathsf{Q}(\textsc{Pointer Chasing}) = \Omega(k)$.

Proof. We can prove this result by a straightforward reduction to parity on $k/2$ bits (let k even), which is known to require $\Omega(k)$ quantum queries [BBC+98]. This reduction is inspired by the well known no-fast forwarding theorem [BACS07]. Let $X \in \{0,1\}^{k/2}$ denote such an instance of parity. We will construct f the first k index/value pairs of f as follows, for all $i \in [0, ..., k/2-1]$.

$$f(2i) = \begin{cases} 2i+2 & \text{if } X_i = 0 \\ 2i+3 & \text{if } X_i = 1 \end{cases} \tag{14}$$

$$f(2i+1) = \begin{cases} 2i+3 & \text{if } X_i = 0 \\ 2i+2 & \text{if } X_i = 1 \end{cases} \tag{15}$$

and set the remaining inputs to be all 0. In this way, we have that $f^{k/2}(0)$ is equal to the parity of X. In particular, there is a path P of length $k/2$ starting at $(0, f(0))$. The i-th link in this path goes from an index of even/odd parity to one of (a) even/odd parity if $X_i = 0$, and (b) odd/even parity if $X_i = 1$. Hence the parity of the full string X is the same as the last bit of $f^k(0)$. □