THEORY OF COMPUTING, Volume 14 (5), 2018, pp. 1–27
www.theoryofcomputing.org

Randomized Query Complexity of Sabotaged and Composed Functions

Shalev Ben-David* Robin Kothari†

Received June 9, 2016; Revised March 2, 2017; Published May 4, 2018

Abstract: We study the composition question for bounded-error randomized query complexity: Is $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(g))$ for all Boolean functions f and g? We show that inserting a simple Boolean function h, whose query complexity is only $\Theta(\log \mathrm{R}(g))$, in between f and g allows us to prove $\mathrm{R}(f \circ h \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(h)\,\mathrm{R}(g))$.

We prove this using a new lower bound measure for randomized query complexity we call randomized sabotage complexity, $\mathrm{RS}(f)$. Randomized sabotage complexity has several desirable properties, such as a perfect composition theorem, $\mathrm{RS}(f \circ g) \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$, and a composition theorem with randomized query complexity, $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{RS}(g))$. It is also a quadratically tight lower bound for total functions and can be quadratically superior to the partition bound, the best known general lower bound for randomized query complexity.

Using this technique we also show implications for lifting theorems in communication complexity. We show that a general lifting theorem for zero-error randomized protocols implies a general lifting theorem for bounded-error protocols.

ACM Classification: F.1.2, F.1.3

AMS Classification: 68W20, 68Q15, 68Q17

Key words and phrases: randomized algorithms, query complexity, composition theorems

A conference version of this paper appeared in the Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016) [6].

*Partially supported by Scott Aaronson's NSF Waterman award under grant number 1249349.
†Partially supported by ARO grant number W911NF-12-1-0486.

 DOI: 10.4086/toc.2018.v014a005 title

1 Introduction

1.1 Composition theorems

A basic structural question that can be asked in any model of computation is whether there can be resource savings when computing the same function on several independent inputs. We say a direct sum theorem holds in a model of computation if solving a problem on n independent inputs requires roughly n times the resources needed to solve one instance of the problem. Direct sum theorems hold for deterministic and randomized query complexity [14], fail for circuit size [21], and remain open for communication complexity [16, 5, 9].

More generally, instead of merely outputting the n answers, we could compute another function of these n answers. If f is an n-bit Boolean function and g is an m-bit Boolean function, we define the composed function $f \circ g$ to be an nm-bit Boolean function such that $f \circ g(x_1, \dots, x_n) = f(g(x_1), \dots, g(x_n))$, where each x_i is an m-bit string. The composition question now asks if there can be significant savings in computing $f \circ g$ compared to simply running the best algorithm for f and using the best algorithm for g to evaluate the input bits needed to compute f. If we let f be the identity function on n bits that just outputs all its inputs, we recover the direct sum problem.

Composition theorems are harder to prove and are known for only a handful of models, such as deterministic [24, 20] and quantum query complexity [22, 19, 17]. More precisely, let $\mathrm{D}(f)$, $\mathrm{R}(f)$, and $\mathrm{Q}(f)$ denote the deterministic, randomized, and quantum query complexities of f. Then for all (possibly partial) Boolean[1] functions f and g, we have

$$\mathrm{D}(f \circ g) = \mathrm{D}(f)\,\mathrm{D}(g) \qquad \text{and} \qquad \mathrm{Q}(f \circ g) = \Theta(\mathrm{Q}(f)\,\mathrm{Q}(g))\,. \tag{1.1}$$

In contrast, in the randomized setting we only have the upper bound $\mathrm{R}(f \circ g) = O(\mathrm{R}(f)\,\mathrm{R}(g) \log \mathrm{R}(f))$. Proving a composition theorem for randomized query complexity remains a major open problem.

Open Problem 1. Does it hold that $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(g))$ for all Boolean functions f and g?

In this paper we prove something close to a composition theorem for randomized query complexity. While we cannot rule out the possibility of synergistic savings in computing $f \circ g$, we show that a composition theorem does hold if we insert a small gadget in between f and g to obfuscate the output of g. Our gadget is "small" in the sense that its randomized (and even deterministic) query complexity is $\Theta(\log \mathrm{R}(g))$. Specifically we choose the index function, which on an input of size $k + 2^k$ interprets the first k bits as an address into the next 2^k bits and outputs the bit stored at that address. The index function's query complexity is $k+1$ and we choose $k = \Theta(\log \mathrm{R}(g))$.

Theorem 1.1. *Let f and g be (partial) Boolean functions and let* $\textsc{Ind}$ *be the index function with* $\mathrm{R}(\textsc{Ind}) = \Theta(\log \mathrm{R}(g))$. *Then*

$$\mathrm{R}(f \circ \textsc{Ind} \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(\textsc{Ind})\,\mathrm{R}(g)) = \Omega(\mathrm{R}(f)\,\mathrm{R}(g) \log \mathrm{R}(g))\,.$$

[1] Composition theorems usually fail for trivial reasons for non-Boolean functions. Hence we restrict our attention to Boolean functions, which have domain $\{0,1\}^n$ (or a subset of $\{0,1\}^n$) and range $\{0,1\}$.

Theorem 1.1 can be used instead of a true composition theorem in many applications. For example, recently a composition theorem for randomized query complexity was needed in the special case when g is the AND function [2]. Our composition theorem would suffice for this application, since the separation shown there only changes by a logarithmic factor if an index gadget is inserted between f and g.

We prove Theorem 1.1 by introducing a new lower bound technique for randomized query complexity. This is not surprising since the composition theorems for deterministic and quantum query complexities are also proved using powerful lower bound techniques for these models, namely, the adversary argument and the negative-weights adversary bound [12], respectively.

1.2 Sabotage complexity

To describe the new lower bound technique, consider the problem of computing a Boolean function f on an input $x \in \{0,1\}^n$ in the query model. In this model we have access to an oracle, which when queried with an index $i \in [n]$ responds with $x_i \in \{0,1\}$.

Imagine that a hypothetical saboteur damages the oracle and makes some of the input bits unreadable. For these input bits the oracle simply responds with a $*$. We can now view the oracle as storing a string $p \in \{0,1,*\}^n$ as opposed to a string $x \in \{0,1\}^n$. Although it is not possible to determine the true input x from the oracle string p, it may still be possible to compute $f(x)$ if all input strings consistent with p evaluate to the same f value. On the other hand, it is not possible to compute $f(x)$ if p is consistent with a 0-input and a 1-input to f. We call such a string $p \in \{0,1,*\}^n$ a *sabotaged input*. For example, let f be the OR function that computes the logical OR of its bits. Then $p = 00{*}0$ is a sabotaged input since it is consistent with the 0-input 0000 and the 1-input 0010. However, $p = 01{*}0$ is not a sabotaged input since it is only consistent with 1-inputs to f.

Now consider a new problem in which the input is promised to be sabotaged (with respect to a function f) and our job is to find the location of a $*$. Intuitively, any algorithm that solves the original problem f when run on a sabotaged input must discover at least one $*$, since otherwise it would answer the same on 0- and 1-inputs consistent with the sabotaged input. Thus the problem of finding a $*$ in a sabotaged input is no harder than the problem of computing f, and hence naturally yields a lower bound on the complexity of computing f. As we show later, this intuition can be formalized in several models of computation.

As it stands the problem of finding a $*$ in a sabotaged input has multiple valid outputs, as the location of any star in the input is a valid output. For convenience we define a decision version of the problem as follows: Imagine there are two saboteurs and one of them has sabotaged our input. The first saboteur, Asterix, replaces input bits with an asterisk ($*$) and the second, Obelix, uses an obelisk (†). Promised that the input has been sabotaged exclusively by one of Asterix or Obelix, our job is to identify the saboteur. This is now a decision problem since there are only two valid outputs. We call this decision problem f_{sab}, the *sabotage problem* associated with f.

We now define lower bound measures for various models using f_{sab}. For example, we can define the *deterministic sabotage complexity of* f as $\mathrm{DS}(f) := \mathrm{D}(f_{\text{sab}})$ and in fact, it turns out that for all f, $\mathrm{DS}(f)$ equals $\mathrm{D}(f)$ (Theorem 8.1).

We could define the *randomized sabotage complexity of* f as $\mathrm{R}(f_{\text{sab}})$, but instead we define it as $\mathrm{RS}(f) := \mathrm{R}_0(f_{\text{sab}})$, where R_0 denotes zero-error randomized query complexity, since $\mathrm{R}(f_{\text{sab}})$ and $\mathrm{R}_0(f_{\text{sab}})$ are equal up to constant factors (Theorem 3.3). $\mathrm{RS}(f)$ has the following desirable properties.

1. (Lower bound for R) For all f, $\mathrm{R}(f) = \Omega(\mathrm{RS}(f))$. (Theorem 3.4)
2. (Perfect composition) For all f and g, $\mathrm{RS}(f \circ g) \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$. (Theorem 4.5)
3. (Composition with R) For all f and g, $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{RS}(g))$. (Theorem 4.7)
4. (Superior to $\mathrm{prt}(f)$) There exists a total f with $\mathrm{RS}(f) \geq \mathrm{prt}(f)^{2-o(1)}$. (Theorem 7.1)
5. (Superior to $\mathrm{Q}(f)$) There exists a total f with $\mathrm{RS}(f) = \widetilde{\Omega}(\mathrm{Q}(f)^{2.5})$. (Theorem 7.1)
6. (Quadratically tight) For all total f, $\mathrm{R}(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$. (Theorem 7.3)

Here $\mathrm{prt}(f)$ denotes the partition bound [13, 15], which subsumes most other lower bound techniques such as approximate polynomial degree, randomized certificate complexity, block sensitivity, etc. The only general lower bound technique not subsumed by $\mathrm{prt}(f)$ is quantum query complexity, $\mathrm{Q}(f)$, which can also be considerably smaller than $\mathrm{RS}(f)$ for some functions. In fact, we are unaware of any total function f for which $\mathrm{RS}(f) = o(\mathrm{R}(f))$, leaving open the intriguing possibility that this lower bound technique captures randomized query complexity for total functions.

1.3 Lifting theorems

Using randomized sabotage complexity we are also able to show a relationship between lifting theorems in communication complexity. A lifting theorem relates the query complexity of a function f with the communication complexity of a related function created from f. Recently, Göös, Pitassi, and Watson [11] showed that there is a communication problem G_{IND}, also known as the two-party index gadget, with communication complexity $\Theta(\log n)$ such that for any function f on n bits, $\mathrm{D}^{\mathrm{cc}}(f \circ G_{\mathrm{IND}}) = \Omega(\mathrm{D}(f)\log n)$, where $\mathrm{D}^{\mathrm{cc}}(F)$ denotes the deterministic communication complexity of a communication problem F.

Analogous lifting theorems are known for some complexity measures, but no such theorem is known for either zero-error randomized or bounded-error randomized query complexity. Our second result shows that a lifting theorem for zero-error randomized query complexity implies one for bounded-error randomized query. We use $\mathrm{R}_0^{\mathrm{cc}}(F)$ and $\mathrm{R}^{\mathrm{cc}}(F)$ to denote the zero-error and bounded-error communication complexities of F, respectively.

Theorem 1.2. *Let $G : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a communication problem with $\min\{|\mathcal{X}|, |\mathcal{Y}|\} = O(\log n)$. If it holds that for all n-bit partial functions f,*

$$\mathrm{R}_0^{\mathrm{cc}}(f \circ G) = \Omega(\mathrm{R}_0(f)/\operatorname{polylog} n)\,, \tag{1.2}$$

then for all n-bit partial functions f,

$$\mathrm{R}^{\mathrm{cc}}(f \circ G_{\mathrm{IND}}) = \Omega(\mathrm{R}(f)/\operatorname{polylog} n)\,, \tag{1.3}$$

where $G_{\mathrm{IND}} : \{0,1\}^b \times \{0,1\}^{2^b} \to \{0,1\}$ is the index gadget (Definition 6.1) with $b = \Theta(\log n)$.

Proving a lifting theorem for bounded-error randomized query complexity remains an important open problem in communication complexity. Such a theorem would allow the recent separations in communication complexity shown by Anshu et al. [4] to be proved simply by establishing their query complexity analogues, which was done in [2] and [3]. Our result shows that it is sufficient to prove a lifting theorem for zero-error randomized protocols instead.

1.4 Open problems

The main open problem is to determine whether $\mathrm{R}(f) = \widetilde{\Theta}(\mathrm{RS}(f))$ for all total functions f. This is known to be false for partial functions, however. Any partial function where all inputs in $\mathrm{Dom}(f)$ are far apart in Hamming distance necessarily has low sabotage complexity. For example, any sabotaged input to the collision problem[2] has at least half the bits sabotaged making $\mathrm{RS}(f) = O(1)$, but $\mathrm{R}(f) = \Omega(\sqrt{n})$.

It would also be interesting to extend the sabotage idea to other models of computation and see if it yields useful lower bound measures. For example, we can define quantum sabotage complexity as $\mathrm{QS}(f) := \mathrm{Q}(f_{\mathrm{sab}})$, but we were unable to show that it lower bounds $\mathrm{Q}(f)$.

1.5 Paper organization

In Section 2, we present some preliminaries and useful properties of randomized algorithms (whose proofs appear in Appendix A for completeness). We then formally define sabotage complexity in Section 3 and prove some basic properties of sabotage complexity. In Section 4 we establish the composition properties of randomized sabotage complexity described above (Theorem 4.5 and Theorem 4.7). Using these results, we establish the main result (Theorem 1.1) in Section 5. We then prove the connection between lifting theorems (Theorem 1.2) in Section 6. In Section 7 we compare randomized sabotage complexity with other lower bound measures. We end with a discussion of deterministic sabotage complexity in Section 8.

2 Preliminaries

In this section we define some basic notions in query complexity that will be used throughout the paper. Note that all the functions in this paper have Boolean input and output, except sabotaged functions whose input alphabet is $\{0,1,*,\dagger\}$. For any positive integer n, we define $[n] := \{1,2,\ldots,n\}$.

In the model of query complexity, we wish to compute an n-bit Boolean function f on an input x given query access to the bits of x. The function f may be total, i. e., $f : \{0,1\}^n \to \{0,1\}$, or partial, which means it is defined only on a subset of $\{0,1\}^n$, which we denote by $\mathrm{Dom}(f)$. The goal is to output $f(x)$ using as few queries to the bits of x as possible. The number of queries used by the best possible deterministic algorithm (over worst-case choice of x) is denoted $\mathrm{D}(f)$.

A randomized algorithm is a probability distribution over deterministic algorithms. The worst-case cost of a randomized algorithm is the worst-case (over all the deterministic algorithms in its support) number of queries made by the algorithm on any input x. The expected cost of the algorithm is the expected number of queries made by the algorithm (over the probability distribution) on an input x maximized over all inputs x. A randomized algorithm has error at most ε if it outputs $f(x)$ on every x with probability at least $1-\varepsilon$.

We use $\mathrm{R}_\varepsilon(f)$ to denote the worst-case cost of the best randomized algorithm that computes f with error ε. Similarly, we use $\overline{\mathrm{R}}_\varepsilon(f)$ to denote the expected cost of the best randomized algorithm that computes f with error ε. When ε is unspecified it is taken to be $\varepsilon = 1/3$. Thus $\mathrm{R}(f)$ denotes the bounded-error randomized query complexity of f. Finally, we also define zero-error expected randomized

[2] In the collision problem, we are given an input $x \in [m]^n$, and we have to decide if x viewed as a function from $[n]$ to $[m]$ is 1-to-1 or 2-to-1 promised that one of these holds.

query complexity, $\overline{\mathrm{R}}_0(f)$, which we also denote by $\mathrm{R}_0(f)$ to be consistent with the literature. For precise definitions of these measures as well as the definition of quantum query complexity $\mathrm{Q}(f)$, see the survey by Buhrman and de Wolf [7].

2.1 Properties of randomized algorithms

We will assume familiarity with the following basic properties of randomized algorithms. For completeness, we prove these properties in Appendix A.

First, we have Markov's inequality, which allows us to convert an algorithm with a guarantee on the expected number of queries into an algorithm with a guarantee on the maximum number of queries with a constant factor loss in the query bound and a constant factor increase in the error. This can be used, for example, to convert zero-error randomized algorithms into bounded-error randomized algorithms.

Lemma 2.1 (Markov's Inequality)**.** *Let A be a randomized algorithm that makes T queries in expectation (over its internal randomness). Then for any $\delta \in (0,1)$, the algorithm A terminates within $\lfloor T/\delta \rfloor$ queries with probability at least $1-\delta$.*

The next property allows us to amplify the success probability of an ε-error randomized algorithm.

Lemma 2.2 (Amplification)**.** *If f is a function with Boolean output and A is a randomized algorithm for f with error $\varepsilon < 1/2$, repeating A several times and taking the majority vote of the outcomes decreases the error. To reach error $\varepsilon' > 0$, it suffices to repeat the algorithm $2\ln(1/\varepsilon')/(1-2\varepsilon)^2$ times.*

Recall that we defined $\overline{\mathrm{R}}_\varepsilon(f)$ to be the minimum expected number of queries made by a randomized algorithm that computes f with error probability at most ε. Clearly, we have $\overline{\mathrm{R}}_\varepsilon(f) \le \mathrm{R}_\varepsilon(f)$, since the expected number of queries made by an algorithm is at most the maximum number of queries made by the algorithm. Using Lemma 2.1, we can now relate them in the other direction.

Lemma 2.3. *Let f be a partial function, $\delta > 0$, and $\varepsilon \in [0,1/2)$. Then we have*

$$\mathrm{R}_{\varepsilon+\delta}(f) \le \frac{1-2\varepsilon}{2\delta}\overline{\mathrm{R}}_\varepsilon(f) \le \frac{1}{2\delta}\overline{\mathrm{R}}_\varepsilon(f)\,.$$

The next lemma shows how to relate these measures with the same error ε on both sides of the inequality. This also shows that $\overline{\mathrm{R}}_\varepsilon(f)$ is only a constant factor away from $\mathrm{R}_\varepsilon(f)$ for constant ε.

Lemma 2.4. *If f is a partial function, then for all $\varepsilon \in (0,1/2)$, we have*

$$\mathrm{R}_\varepsilon(f) \le 14\frac{\ln(1/\varepsilon)}{(1-2\varepsilon)^2}\overline{\mathrm{R}}_\varepsilon(f)\,.$$

When $\varepsilon = 1/3$, we can improve this to $\mathrm{R}(f) \le 10\overline{\mathrm{R}}(f)$.

Although these measures are closely related for constant error, $\overline{\mathrm{R}}_\varepsilon(f)$ is more convenient than $\mathrm{R}_\varepsilon(f)$ for discussing composition and direct sum theorems.

We can also convert randomized algorithms that find certificates with bounded error into zero-error randomized algorithms.

Lemma 2.5. *Let A be a randomized algorithm that uses T queries in expectation and finds a certificate with probability $1-\varepsilon$. Then repeating A when it fails to find a certificate turns it into an algorithm that always finds a certificate (i. e., a zero-error algorithm) that uses at most $T/(1-\varepsilon)$ queries in expectation.*

Finally, the following lemma is useful for proving lower bounds on randomized algorithms.

Lemma 2.6. *Let f be a partial function. Let A be a randomized algorithm that solves f using at most T expected queries and with error at most ε. For $x,y \in \mathrm{Dom}(f)$ if $f(x) \neq f(y)$ then when A is run on x, it must query an entry on which x differs from y with probability at least $1-2\varepsilon$.*

3 Sabotage complexity

We now formally define sabotage complexity. Given a (partial or total) n-bit Boolean function f, let $P_f \subseteq \{0,1,*\}^n$ be the set of all partial assignments of f that are consistent with both a 0-input and a 1-input. That is, for each $p \in P_f$, there exist $x,y \in \mathrm{Dom}(f)$ such that $f(x) \neq f(y)$ and $x_i = y_i = p_i$ whenever $p_i \neq *$. Let $P_f^\dagger \subseteq \{0,1,\dagger\}^n$ be the same as P_f, except using the symbol $\dagger$ instead of $*$. Observe that P_f and $P_f^\dagger$ are disjoint. Let $Q_f = P_f \cup P_f^\dagger \subseteq \{0,1,*,\dagger\}^n$. We then define f_{sab} as follows.

Definition 3.1. Let f be an n-bit partial function. We define $f_{\mathrm{sab}} : Q_f \to \{0,1\}$ as $f_{\mathrm{sab}}(q) = 0$ if $q \in P_f$ and $f_{\mathrm{sab}}(q) = 1$ if $q \in P_f^\dagger$.

Note that even when f is a total function, f_{sab} is always a partial function. See Section 1.2 for more discussion and motivation for this definition. Now that we have defined f_{sab}, we can define deterministic and randomized sabotage complexity.

Definition 3.2. Let f be a partial function. Then $\mathrm{DS}(f) := \mathrm{D}(f_{\mathrm{sab}})$ and $\mathrm{RS}(f) := \mathrm{R}_0(f_{\mathrm{sab}})$.

We will primarily focus on $\mathrm{RS}(f)$ in this work and only discuss $\mathrm{DS}(f)$ in Section 8. To justify defining $\mathrm{RS}(f)$ as $\mathrm{R}_0(f_{\mathrm{sab}})$ instead of $\mathrm{R}(f_{\mathrm{sab}})$ (or $\overline{\mathrm{R}}(f_{\mathrm{sab}})$), we now show these definitions are equivalent up to constant factors.

Theorem 3.3. *Let f be a partial function. Then* $\mathrm{R}_0(f_{\mathrm{sab}}) \geq \overline{\mathrm{R}}_\varepsilon(f_{\mathrm{sab}}) \geq (1-2\varepsilon)\mathrm{R}_0(f_{\mathrm{sab}})$.

Proof. The first inequality follows trivially. For the second, let $x \in Q_f$ be any valid input to f_{sab}. Let x' be the input x with asterisks replaced with obelisks and vice versa. Then since $f_{\mathrm{sab}}(x) \neq f_{\mathrm{sab}}(x')$, by Lemma 2.6 any ε-error randomized algorithm that solves f_{sab} must find a position on which x and x' differ with probability at least $1-2\varepsilon$. The positions at which they differ are either asterisks or obelisks. Since x was an arbitrary input, the algorithm must always find an asterisk or obelisk with probability at least $1-2\varepsilon$. Since finding an asterisk or obelisk is a certificate for f_{sab}, by Lemma 2.5, we get a zero-error algorithm for f_{sab} that uses $\overline{\mathrm{R}}_\varepsilon(f_{\mathrm{sab}})/(1-2\varepsilon)$ expected queries. Thus $\mathrm{R}_0(f_{\mathrm{sab}}) \leq \overline{\mathrm{R}}_\varepsilon(f_{\mathrm{sab}})/(1-2\varepsilon)$, as desired. □

Finally, we prove that $\mathrm{RS}(f)$ is indeed a lower bound on $\mathrm{R}(f)$, i. e., $\mathrm{R}(f) = \Omega(\mathrm{RS}(f))$.

Theorem 3.4. *Let f be a partial function. Then* $\mathrm{R}_\varepsilon(f) \geq \overline{\mathrm{R}}_\varepsilon(f) \geq (1-2\varepsilon)\mathrm{RS}(f)$.

Proof. Let A be a randomized algorithm for f that uses $\overline{\mathrm{R}}_\varepsilon(f)$ randomized queries and outputs the correct answer on every input in $\mathrm{Dom}(f)$ with probability at least $1-\varepsilon$. Now fix a sabotaged input x, and let p be the probability that A finds a $*$ or $\dagger$ when run on x. Let q be the probability that A outputs 0 if it doesn't find a $*$ or $\dagger$ when run on x. Let x_0 and x_1 be inputs consistent with x such that $f(x_0)=0$ and $f(x_1)=1$. Then A outputs 0 on x_1 with probability at least $q(1-p)$, and A outputs 1 on x_0 with probability at least $(1-q)(1-p)$. These are both errors, so we have $q(1-p)\le\varepsilon$ and $(1-q)(1-p)\le\varepsilon$. Summing them gives $1-p\le 2\varepsilon$, or $p\ge 1-2\varepsilon$.

This means A finds a $*$ entry within $\overline{\mathrm{R}}_\varepsilon(f)$ expected queries with probability at least $1-2\varepsilon$. By Lemma 2.5, we get

$$\frac{1}{1-2\varepsilon}\overline{\mathrm{R}}_\varepsilon(f)\ge \mathrm{RS}(f)\,,\qquad \text{or}\qquad \overline{\mathrm{R}}_\varepsilon(f)\ge(1-2\varepsilon)\,\mathrm{RS}(f)\,. \tag{3.1}$$

□

We also define a variant of RS where the number of asterisks (or obelisks) is limited to one. Specifically, let $U\subseteq\{0,1,*,\dagger\}^n$ be the set of all partial assignments with exactly one $*$ or $\dagger$. Formally,

$$U:=\left\{x\in\{0,1,*,\dagger\}^n : |\{i\in[n]:x_i\notin\{0,1\}\}|=1\right\}. \tag{3.2}$$

Definition 3.5. Let f be an n-bit partial function. We define f_{usab} as the restriction of f_{sab} to U, the set of strings with only one asterisk or obelisk. That is, f_{usab} has domain $Q_f\cap U$, but is equal to f_{sab} on its domain. We then define $\mathrm{RS_u}(f):=\mathrm{R}_0(f_{\mathrm{usab}})$. If $Q_f\cap U$ is empty, we define $\mathrm{RS_u}(f):=0$.

The measure $\mathrm{RS_u}$ will play a key role in our lifting result in Section 6. Since f_{usab} is a restriction of f_{sab} to a promise, it is clear that its zero-error randomized query complexity cannot increase, and so $\mathrm{RS_u}(f)\le\mathrm{RS}(f)$. Interestingly, when f is total, $\mathrm{RS_u}(f)$ equals $\mathrm{RS}(f)$. In other words, when f is total, we may assume without loss of generality that its sabotaged version has only one asterisk or obelisk.

Theorem 3.6. *If f is a total function, then* $\mathrm{RS_u}(f)=\mathrm{RS}(f)$.

Proof. We already argued that $\mathrm{RS}(f)\ge\mathrm{RS_u}(f)$. To show $\mathrm{RS_u}(f)\ge\mathrm{RS}(f)$, we argue that any zero-error algorithm A for f_{usab} also solves f_{sab}. The main observation we need is that any input to f_{sab} can be completed to an input to f_{usab} by replacing some asterisks or obelisks with 0s and 1s. To see this, let x be an input to f_{sab}. Without loss of generality, $x\in P_f$. Then there are two strings $y,z\in\mathrm{Dom}(f)$ that are consistent with x, satisfying $f(y)=0$ and $f(z)=1$.

The strings y and z disagree on some set of bits B, and x has a $*$ or $\dagger$ on all of B. Consider starting with y and flipping the bits of B one by one, until we reach the string z. At the beginning, we have $f(y)=0$, and at the end, we reach $f(z)=1$. This means that at some point in the middle, we must have flipped a bit that flipped the string from a 0-input to a 1-input. Let w_0 and w_1 be the inputs where this happens. They differ in only one bit. If we replace that bit with $*$ or $\dagger$, we get a partial assignment w consistent with both, so $w\in P_f$. Moreover, w is consistent with x. This means we have completed an arbitrary input to f_{sab} to an input to f_{usab}, as claimed.

The algorithm A, which correctly solves f_{usab}, when run on w (a valid input to f_{usab}) must find an asterisk or obelisk in w. Now consider running A on the input x to f_{sab} and compare its execution to when it is run on w. If A ever queries a position that is different in x and w, then it has found an asterisk or obelisk and the algorithm can now halt. If not, then it must find the single asterisk or obelisk present in w, which is also present in x. This shows that the slightly modified version of A that halts if it queries an asterisk or obelisk solves f_{sab} and hence $\mathrm{RS}(f)=\mathrm{R}_0(f_{\mathrm{sab}})\le\mathrm{R}_0(f_{\mathrm{usab}})=\mathrm{RS_u}(f)$. □

4 Direct sum and composition theorems

In this section, we establish the main composition theorems for randomized sabotage complexity. To do so, we first need to establish direct sum theorems for the problem f_{sab}. In fact, our direct sum theorems hold more generally for zero-error randomized query complexity of partial functions (and even relations). To prove this, we will require Yao's minimax theorem [27].

Theorem 4.1 (Minimax)**.** *Let f be an n-bit partial function. There is a distribution μ over inputs in* $\mathrm{Dom}(f)$ *such that all zero-error algorithms for f use at least* $\mathrm{R}_0(f)$ *expected queries on μ.*

We call any distribution μ that satisfies the assertion in Yao's theorem a *hard distribution* for f.

4.1 Direct sum theorems

We start by defining the m-fold direct sum of a function f, which is simply the function that accepts m inputs to f and outputs f evaluated on all of them.

Definition 4.2. Let $f : \mathrm{Dom}(f) \to \mathcal{Z}$, where $\mathrm{Dom}(f) \subseteq \mathcal{X}^n$, be a partial function with input and output alphabets $\mathcal{X}$ and $\mathcal{Z}$. The m-fold direct sum of f is the partial function $f^{\oplus m} : \mathrm{Dom}(f)^m \to \mathcal{Z}^m$ such that for any $(x_1, x_2, \ldots, x_m) \in \mathrm{Dom}(f)^m$, we have

$$f^{\oplus m}(x_1, x_2, \ldots, x_m) = (f(x_1), f(x_2), \ldots, f(x_m)). \tag{4.1}$$

We can now prove a direct sum theorem for zero-error randomized and more generally ε-error expected randomized query complexity, although we only require the result about zero-error algorithms. We prove these results for partial functions, but they also hold for arbitrary relations.

Theorem 4.3 (Direct sum)**.** *For any n-bit partial function f and any positive integer m, we have* $\mathrm{R}_0(f^{\oplus m}) = m\mathrm{R}_0(f)$*. Moreover, if μ is a hard distribution for f given by Theorem 4.1, then $\mu^{\otimes m}$ is a hard distribution for $f^{\oplus m}$. Similarly, for ε-error randomized algorithms we get* $\overline{\mathrm{R}}_\varepsilon(f^{\oplus m}) \geq m\overline{\mathrm{R}}_\varepsilon(f)$.

Proof. The upper bound follows from running the $\mathrm{R}_0(f)$ algorithm on each of the m inputs to f. By linearity of expectation, this algorithm solves all m inputs after $m\mathrm{R}_0(f)$ expected queries.

We now prove the lower bound. Let A be a zero-error randomized algorithm for $f^{\oplus m}$ that uses T expected queries when run on inputs from $\mu^{\otimes m}$. We convert A into an algorithm B for f that uses T/m expected queries when run on inputs from μ.

Given an input $x \sim \mu$, the algorithm B generates $m-1$ additional "fake" inputs from μ. B then shuffles these together with x, and runs A on the result. The input to A is then distributed according to $\mu^{\otimes m}$, so A uses T queries (in expectation) to solve all m inputs. B then reads the solution to the true input x.

Note that most of the queries A makes are to fake inputs, so they don't count as real queries. The only real queries B has to make happen when A queries x. But since x is shuffled with the other (indistinguishable) inputs, the expected number of queries A makes to x is the same as the expected number of queries A makes to each fake input; this must equal T/m. Thus B makes T/m queries to x (in expectation) before solving it.

Since B is a zero-error randomized algorithm for f that uses T/m expected queries on inputs from μ, we must have $T/m \geq \mathrm{R}_0(f)$ by Theorem 4.1. Thus $T \geq m\,\mathrm{R}_0(f)$, as desired.

The same lower bound proof carries through for ε-error expected query complexity, $\overline{\mathrm{R}}_\varepsilon(f)$, as long as we use a version of Yao's theorem for this model. For completeness, we prove this version of Yao's theorem in Appendix B. □

Theorem 4.3 is essentially [14, Theorem 2], but our theorem statement looks different since we deal with expected query complexity instead of worst-case query complexity. From Theorem 4.3, we can also prove a direct sum theorem for worst-case randomized query complexity since for $\varepsilon \in (0,1/2)$,

$$\mathrm{R}_\varepsilon(f^{\oplus m}) \geq \overline{\mathrm{R}}_\varepsilon(f^{\oplus m}) \geq m\overline{\mathrm{R}}_\varepsilon(f) \geq 2\delta m\,\mathrm{R}_{\varepsilon+\delta}(f)\,, \tag{4.2}$$

for any $\delta > 0$, where the last inequality used Lemma 2.3.

For our applications, however, we will need a strengthened version of this theorem, which we call a threshold direct sum theorem.

Theorem 4.4 (Threshold direct sum)**.** *Given an input to $f^{\oplus m}$ sampled from $\mu^{\otimes m}$, we consider solving only some of the m inputs to f. We say an input x to f is solved if a z-certificate was queried that proves $f(x) = z$. Then any randomized algorithm that takes an expected T queries and solves an expected k of the m inputs when run on inputs from $\mu^{\otimes m}$ must satisfy $T \geq k\,\mathrm{R}_0(f)$.*

Proof. We prove this by a reduction to Theorem 4.3. Let A be a randomized algorithm that, when run on an input from $\mu^{\otimes m}$, solves an expected k of the m instances, and halts after an expected T queries. We note that these expectations average over both the distribution $\mu^{\otimes m}$ and the internal randomness of A.

We now define a randomized algorithm B that solves the m-fold direct sum $f^{\oplus m}$ with zero error. B works as follows: given an input to $f^{\oplus m}$, B first runs A on that input. Then B checks which of the m instances of f were solved by A (by seeing if a certificate proving the value of f was found for a given instance of f). B then runs the optimal zero-error algorithm for f, which makes $\mathrm{R}_0(f)$ expected queries, on the instances of f that were not solved by A.

Let us examine the expected number of queries used by B on an input from $\mu^{\otimes m}$. Recall that a randomized algorithm is a probability distribution over deterministic algorithms; we can therefore think of A as a distribution. For a deterministic algorithm $D \sim A$ and an input x to $f^{\oplus m}$, we use $D(x)$ to denote the number of queries used by D on x, and $S(D,x) \subseteq [m]$ to denote the set of inputs to f the algorithm D solves when run on x. Then by assumption

$$T = \mathop{\mathbb{E}}_{x\sim\mu^{\otimes m}} \mathop{\mathbb{E}}_{D\sim A} D(x) \qquad \text{and} \qquad k = \mathop{\mathbb{E}}_{x\sim\mu^{\otimes m}} \mathop{\mathbb{E}}_{D\sim A} |S(D,x)|\,. \tag{4.3}$$

Next, let R be the randomized algorithm that uses $\mathrm{R}_0(f)$ expected queries and solves f on any input. For an input x to $f^{\oplus m}$, we write $x = x_1x_2\ldots x_m$ with $x_i \in \mathrm{Dom}(f)$. Then the expected number of queries used by B on input from $\mu^{\otimes m}$ can be written as

$$\mathop{\mathbb{E}}_{x\sim\mu^{\otimes m}} \mathop{\mathbb{E}}_{D\sim A} \left(D(x) + \mathop{\mathbb{E}}_{D_1\sim R} \mathop{\mathbb{E}}_{D_2\sim R} \cdots \mathop{\mathbb{E}}_{D_m\sim R} \sum_{i\in[m]\setminus S(D,x)} D_i(x_i) \right) \tag{4.4}$$

$$= \mathop{\mathbb{E}}_{x\sim\mu^{\otimes m}} \mathop{\mathbb{E}}_{D\sim A} \left(D(x) + \sum_{i\in[m]\setminus S(D,x)} \mathop{\mathbb{E}}_{D_i\sim R} D_i(x_i) \right) \tag{4.5}$$

$$\leq \mathop{\mathbb{E}}_{x\sim\mu^{\otimes m}} \mathop{\mathbb{E}}_{D\sim A} \left(D(x) + \sum_{i\in[m]\setminus S(D,x)} \mathrm{R}_0(f) \right) \tag{4.6}$$

$$= \mathop{\mathbb{E}}_{x\sim\mu^{\otimes m}} \mathop{\mathbb{E}}_{D\sim A} \left(D(x) + (m - |S(D,x)|)\,\mathrm{R}_0(f) \right) \tag{4.7}$$

$$= \quad T + (m-k)\,\mathrm{R}_0(f)\,. \tag{4.8}$$

Since B solves the direct sum problem on $\mu^{\otimes m}$, the expected number of queries it uses is at least $m\mathrm{R}_0(f)$ by Theorem 4.3. Hence $T + (m-k)\mathrm{R}_0(f) \geq m\mathrm{R}_0(f)$, so $T \geq k\mathrm{R}_0(f)$. □

4.2 Composition theorems

Using the direct sum and threshold direct sum theorems we have established, we can now prove composition theorems for randomized sabotage complexity. We start with the behavior of RS itself under composition.

Theorem 4.5. *Let f and g be partial functions. Then* $\mathrm{RS}(f\circ g) \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$.

Proof. Let A be any zero-error algorithm for $(f\circ g)_{\mathrm{sab}}$, and let T be the expected query complexity of A (maximized over all inputs). We turn A into a zero-error algorithm B for f_{sab}.

B takes a sabotaged input x for f. It then runs A on a sabotaged input to $f\circ g$ constructed as follows. Each 0 bit of x is replaced with a 0-input to g, each 1 bit of x is replaced with a 1-input to g, and each $*$ or $\dagger$ of x is replaced with a sabotaged input to g. The sabotaged inputs are generated from μ, the hard distribution for g_{sab} obtained from Theorem 4.1. The 0-inputs are generated by first generating a sabotaged input, and then selecting a 0-input consistent with that sabotaged input. The 1-inputs are generated analogously.

This is implemented in the following way. On input x, the algorithm B generates n sabotaged inputs from μ (the hard distribution for g_{sab}), where n is the length of the string x. Call these inputs $y_1, y_2, \ldots, y_n$. B then runs the algorithm A on this collection of n strings, pretending that it is an input to $f\circ g$, with the following caveat: whenever A tries to query a $*$ or $\dagger$ in an input y_i, B instead queries x_i. If x_i is 0, B selects an input from $f^{-1}(0)$ consistent with y_i, and replaces y_i with this input. It then returns to A an answer consistent with the new y_i. If x_i is 1, B selects a consistent input from $f^{-1}(1)$ instead. If x_i is a $*$ or $\dagger$, B returns a $*$ or $\dagger$, respectively.

Now B only makes queries to x when it finds a $*$ or $\dagger$ in an input to g_{sab}. But this solves that instance of g_{sab}, which was drawn from the hard distribution for g_{sab}. Thus the query complexity of B is upper bounded by the number of instances of g_{sab} that can be solved by a T-query algorithm with access to n instances of g_{sab}. We know from Theorem 4.4 that if A makes T expected queries, the expected number of $*$ or $\dagger$ entries it finds among $y_1, y_2, \ldots, y_n$ is at most $T/\mathrm{RS}(g)$. Hence the expected number of queries B makes to x is at most $T/\mathrm{RS}(g)$. Thus we have $\mathrm{RS}(f) \le T/\mathrm{RS}(g)$, which gives $T \ge \mathrm{RS}(f)\,\mathrm{RS}(g)$. $\square$

Using this we can lower bound the randomized query complexity of composed functions. In the following, f^n denotes the function f composed with itself n times, i. e., $f^1 = f$ and $f^{i+1} = f \circ f^i$.

Corollary 4.6. *Let f be a partial function. Then* $\mathrm{R}(f^n) \ge \mathrm{RS}(f)^n/3$.

This follows straightforwardly from observing that $\mathrm{R}(f^n) = \mathrm{R}_{1/3}(f^n) \ge (1-2/3)\,\mathrm{RS}(f^n)$ (using Theorem 3.4) and $\mathrm{RS}(f^n) \ge \mathrm{RS}(f)^n$ (using Theorem 4.5).

We can also prove a composition theorem for zero-error and bounded-error randomized query complexity in terms of randomized sabotage complexity. In particular this yields a composition theorem for $\mathrm{R}(f \circ g)$ when $\mathrm{R}(g) = \Theta(\mathrm{RS}(g))$.

Theorem 4.7. *Let f and g be partial functions. Then* $\overline{\mathrm{R}}_\varepsilon(f \circ g) \ge \overline{\mathrm{R}}_\varepsilon(f)\,\mathrm{RS}(g)$.

Proof. The proof follows a similar argument to the proof of Theorem 4.5. Let A be a randomized algorithm for $f \circ g$ that uses T expected queries and makes error ε. We turn A into an algorithm B for f by having B generate inputs from μ, the hard distribution for g_{sab}, and feeding them to A, as before. The only difference is that this time, the input x to B is not a sabotaged input. This means it has no $*$ or $\dagger$ entries, so all the sabotaged inputs that B generates turn into 0- or 1-inputs if A tries to query a $*$ or $\dagger$ in them.

Since A uses T queries, by Theorem 4.4, it finds at most $T/\mathrm{RS}(g)$ asterisks or obelisks (in expectation). Therefore, B makes at most $T/\mathrm{RS}(g)$ expected queries to x. Since B is correct whenever A is correct, its error probability is at most ε. Thus $\overline{\mathrm{R}}_\varepsilon(f) \le T/\mathrm{RS}(g)$, and thus $T \ge \overline{\mathrm{R}}_\varepsilon(f)\,\mathrm{RS}(g)$. $\square$

Setting ε to 0 yields the following corollary.

Corollary 4.8. *Let f and g be partial functions. Then* $\mathrm{R}_0(f \circ g) \ge \mathrm{R}_0(f)\,\mathrm{RS}(g)$.

For the more commonly used $\mathrm{R}(f \circ g)$, we obtain the following composition result.

Corollary 4.9. *Let f and g be partial functions. Then* $\mathrm{R}(f \circ g) \ge \mathrm{R}(f)\,\mathrm{RS}(g)/10$.

This follows from Lemma 2.4, which gives $\overline{\mathrm{R}}_{1/3}(f) \ge \mathrm{R}(f)/10$, and Theorem 4.7, since

$$\mathrm{R}(f \circ g) \ge \overline{\mathrm{R}}_{1/3}(f \circ g) \ge \overline{\mathrm{R}}_{1/3}(f)\,\mathrm{RS}(g) \ge \mathrm{R}(f)\,\mathrm{RS}(g)/10\,. \tag{4.9}$$

Finally, we can also show an upper bound composition result for randomized sabotage complexity.

Theorem 4.10. *Let f and g be partial functions. Then* $\mathrm{RS}(f \circ g) \le \mathrm{RS}(f)\,\mathrm{R}_0(g)$. *We also have*

$$\mathrm{RS}(f \circ g) = O\big(\mathrm{RS}(f)\,\mathrm{R}(g) \log \mathrm{RS}(f)\big)\,.$$

Proof. We describe a simple algorithm for finding a $*$ or $\dagger$ in an input to $f \circ g$. Start by running the optimal algorithm for the sabotage problem of f. This algorithm uses $\mathrm{RS}(f)$ expected queries. Then whenever this algorithm tries to query a bit, run the optimal zero-error algorithm for g in the corresponding input to g.

Now, since the input to $f \circ g$ that we are given is a sabotaged input, it must be consistent with both a 0-input and a 1-input of $f \circ g$. It follows that some of the g inputs are sabotaged, and moreover, if we represent a sabotaged g-input by $*$ or $\dagger$, a 0-input to g by 0, and a 1-input to g by 1, we get a sabotaged input to f. In other words, from the inputs to g we can derive a sabotaged input for f.

This means that the outer algorithm runs uses an expected $\mathrm{RS}(f)$ calls to the inner algorithm, and ends up calling the inner algorithm on a sabotaged input to g. Meanwhile, each call to the inner algorithm uses an expected $\mathrm{R}_0(g)$ queries, and will necessarily find a $*$ or $\dagger$ if the input it is run on is sabotaged. Therefore, the described algorithm will always find a $*$ or $\dagger$, and its expected running time is $\mathrm{RS}(f)\,\mathrm{R}_0(g)$ by linearity of expectation and by the independence of the internal randomness of the two algorithms.

Instead of using a zero-error randomized algorithm for g, we can use a bounded-error randomized algorithm for g as long as its error probability is small. Since we make $O(\mathrm{RS}(f))$ calls to the inner algorithm, if we boost the bounded-error algorithm's success probability to make the error much smaller than $1/\mathrm{RS}(f)$ (costing an additional $\log \mathrm{RS}(f)$ factor), we will get a bounded-error algorithm for $(f \circ g)_{\mathrm{sab}}$. Since $\mathrm{R}((f \circ g)_{\mathrm{sab}})$ is the same as $\mathrm{RS}(f \circ g)$ up to a constant factor (Theorem 3.3),

$$\mathrm{RS}(f \circ g) = O(\mathrm{RS}(f)\,\mathrm{R}(g) \log \mathrm{RS}(f)), \tag{4.10}$$

as desired. ☐

5 Composition with the index function

We now prove our main result (Theorem 1.1) restated more precisely as follows.

Theorem 1.1 (Precise version)**.** *Let f and g be (partial) functions, and let $m = \Omega(\mathrm{R}(g)^{1.1})$. Then*

$$\mathrm{R}(f \circ \mathrm{IND}_m \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(g) \log m) = \Omega(\mathrm{R}(f)\,\mathrm{R}(\mathrm{IND}_m)\,\mathrm{R}(g)).$$

Before proving this, we formally define the index function.

Definition 5.1 (Index function)**.** The index function on m bits, denoted $\mathrm{IND}_m : \{0,1\}^m \to \{0,1\}$, is defined as follows. Let c be the largest integer such that $c + 2^c \le m$. For any input $x \in \{0,1\}^m$, let y be the first c bits of x and let $z = z_0 z_1 \cdots z_{2^c-1}$ be the next 2^c bits of x. If we interpret y as the binary representation of an integer between 0 and $2^c - 1$, then the output of $\mathrm{IND}_m(x)$ equals z_y.

To prove Theorem 1.1, we also require the strong direct product theorem for randomized query complexity that was established by Drucker [8].

Theorem 5.2 (Strong direct product)**.** *Let f be a partial Boolean function, and let k be a positive integer. Then any randomized algorithm for $f^{\oplus k}$ that uses at most $\gamma^3 k \mathrm{R}(f)/11$ queries has success probability at most $(1/2 + \gamma)^k$, for any $\gamma \in (0, 1/4)$.*

The first step to proving $\mathrm{R}(f \circ \textsc{Ind} \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(\textsc{Ind})\,\mathrm{R}(g))$ is to establish that $\mathrm{R}(\textsc{Ind} \circ g)$ is essentially the same as $\mathrm{RS}(\textsc{Ind} \circ g)$ if the index gadget is large enough.

Lemma 5.3. *Let f be a partial Boolean function and let $m = \Omega(\mathrm{R}(f)^{1.1})$. Then*

$$\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(\mathrm{R}(f)\log m) = \Omega(\mathrm{R}(\textsc{Ind}_m)\,\mathrm{R}(f))\,. \tag{5.1}$$

Moreover, if $f_{\mathrm{ind}}^{\oplus c}$ is defined as the index function on $c+2^c$ bits composed with f in only the first c bits, we have $\mathrm{RS}(f_{\mathrm{ind}}^{\oplus c}) \geq \mathrm{RS}_{\mathrm{u}}(f_{\mathrm{ind}}^{\oplus c}) = \Omega(c\,\mathrm{R}(f))$ when $c \geq 1.1\log \mathrm{R}(f)$.

Before proving Lemma 5.3, let us complete the proof of Theorem 1.1 assuming Lemma 5.3.

Proof of Theorem 1.1. By Corollary 4.9, we have $\mathrm{R}(f \circ \textsc{Ind}_m \circ g) \geq \mathrm{R}(f)\,\mathrm{RS}(\textsc{Ind}_m \circ g)/10$. Combining this with Lemma 5.3 gives $\mathrm{R}(f \circ \textsc{Ind}_m \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(g)\log m)$, as desired. □

We can now complete the argument by proving Lemma 5.3.

Proof of Lemma 5.3. To understand what the inputs to $(\textsc{Ind}_m \circ f)_{\mathrm{sab}}$ look like, let us first analyze the function $\textsc{Ind}_m$. We can split an input to $\textsc{Ind}_m$ into a small index section and a large array section. To sabotage an input to $\textsc{Ind}_m$, it suffices to sabotage the array element that the index points to (using only a single $*$ or $\dagger$). It follows that to sabotage an input to $\textsc{Ind}_m \circ f$, it suffices to sabotage the input to f at the array element that the index points to. In other words, we consider sabotaged inputs where the only stars in the input are in one array cell whose index is the output of the first c copies of f, where c is the largest integer such that $c + 2^c \leq m$. Note that $c = \log m - \Theta(1)$.

We now convert any $\mathrm{RS}(\textsc{Ind}_m \circ f)$ algorithm into a randomized algorithm for $f^{\oplus c}$. First, using Lemma 2.1, we get a $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ query randomized algorithm that finds a $*$ or $\dagger$ with probability $1/2$ if the input is sabotaged. Next, consider running this algorithm on a non-sabotaged input. It makes $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries. With probability $1/2$, one of these queries will be in the array cell whose index is the true answer to $f^{\oplus c}$ evaluated on the first cn bits. We can then consider a new algorithm A that runs the above algorithm for $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries, then picks one of the $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries at random, and if that query is in an array cell, it outputs the index of that cell. Then A uses $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries and evaluates $f^{\oplus c}$ with probability at least $\mathrm{RS}(\textsc{Ind}_m \circ f)^{-1}/4$.

Next, Theorem 5.2 implies that for any $\gamma \in (0, 1/4)$, either A's success probability is smaller than $(1/2+\gamma)^c$, or else A uses at least $\gamma^3 c\,\mathrm{R}(f)/11$ queries. This means either

$$\mathrm{RS}(\textsc{Ind}_m \circ f)^{-1}/4 \leq (1/2+\gamma)^c \qquad \text{or} \qquad 2\,\mathrm{RS}(\textsc{Ind}_m \circ f) \geq \gamma^3 c\,\mathrm{R}(f)/11\,. \tag{5.2}$$

Now if we choose $\gamma = 0.01$, it is clear that the second inequality in (5.2) yields $\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(c\,\mathrm{R}(f)) = \Omega(\mathrm{R}(f)\log m)$ no matter what m (and hence c) is chosen to be.

To complete the argument, we show that the first inequality in (5.2) also yields the same. Observe that the first inequality is equivalent to

$$\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega\left(\left(\frac{2}{1+2\gamma}\right)^c\right) = \Omega\left(\left(\frac{2}{1+2\gamma}\right)^{\log m - \Theta(1)}\right) = \Omega(m^{\log_2(2/1.02)}) = \Omega(m^{0.97})\,. \tag{5.3}$$

We now have $m^{0.97} = \Omega(m^{0.96}\log m) = \Omega(\mathrm{R}(f)^{1.1\times 0.96}\log m) = \Omega(\mathrm{R}(f)\log m)$, as desired.

The lower bound on $\mathrm{RS_u}(f_{\mathrm{ind}}^{\oplus c})$ follows similarly once we makes two observations. First, this argument works equally well for $f_{\mathrm{ind}}^{\oplus c}$ instead of $\mathrm{IND}_m \circ f$. Second, sabotaging the array cell indexed by the outputs to the c copies of f in $f_{\mathrm{ind}}^{\oplus c}$ introduces only one asterisk or obelisk, so the argument above lower bounds $\mathrm{RS_u}(f_{\mathrm{ind}}^{\oplus c})$ and not only $\mathrm{RS}(f_{\mathrm{ind}}^{\oplus c})$. □

6 Relating lifting theorems

In this section we establish Theorem 1.2, which proves that a lifting theorem for zero-error randomized communication complexity implies one for bounded-error randomized communication complexity.

To begin, we introduce the two-party index gadget (also used in [11]).

Definition 6.1 (Two-party index gadget)**.** For any integer $b > 0$, and finite set $\mathcal{Y}$, we define the index function $G_{\mathrm{IND}} : \{0,1\}^b \times \mathcal{Y}^{2^b} \to \mathcal{Y}$ as follows. Let $(x,y) \in \{0,1\}^b \times \mathcal{Y}^{2^b}$ be an input to G_{IND}. Then if we interpret x as the binary representation of an integer between 0 and $2^b - 1$, the function $G_{\mathrm{IND}}(x,y)$ evaluates to y_x, the x^{th} letter of y. We also let G_b be the index function with $\mathcal{Y} = \{0,1\}$ and let G'_b be the index function with $\mathcal{Y} = \{0,1,*,\dagger\}$.

The index gadget is particularly useful in communication complexity because it is "complete" for functions with a given value of $\min\{|\mathcal{X}|,|\mathcal{Y}|\}$. More precisely, any problem $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ can be reduced to G_b for $b = \lceil \log \min\{|\mathcal{X}|,|\mathcal{Y}|\} \rceil$. To see this, say $|\mathcal{X}| \le |\mathcal{Y}|$ and let $|\mathcal{X}| = 2^b$. We now map every input $(x,y) \in \mathcal{X} \times \mathcal{Y}$ to an input (x',y') for G_b. Since $\mathcal{X}$ has size 2^b, we can view x as a string in $\{0,1\}^b$ and set $x' = x$. The string $y' = y'_0 y'_1 \cdots y'_{2^b-1} \in \{0,1\}^{2^b}$ is defined as $y'_x = F(x,y)$. Hence we can assume without loss of generality that a supposed lifting theorem for zero-error protocols is proved using the two-party index gadget of some size.

Our first step is to lower bound the bounded-error randomized communication complexity of a function in terms of the zero-error randomized communication complexity of a related function.

Lemma 6.2. *Let f be an n-bit (partial) Boolean function and let $G_b : \{0,1\}^b \times \{0,1\}^{2^b} \to \{0,1\}$ be the index gadget with $b = O(\log n)$. Then*

$$\mathrm{R}^{\mathrm{cc}}(f \circ G_b) = \Omega\left(\frac{\mathrm{R}_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b)}{\log n \log\log n}\right), \tag{6.1}$$

where G'_b is the index gadget mapping $\{0,1\}^b \times \{0,1,*,\dagger\}^{2^b}$ to $\{0,1,*,\dagger\}$.

Proof. We will use a randomized protocol A for $f \circ G_b$ to construct a zero-error protocol B for $f_{\mathrm{usab}} \circ G'_b$. Note the given input to $f_{\mathrm{usab}} \circ G'_b$ must have a unique copy of G'_b that evaluates to $*$ or $\dagger$, with all other copies evaluating to 0 or 1. The goal of B is to find this copy and determine if it evaluates to $*$ or $\dagger$. This will evaluate $f_{\mathrm{usab}} \circ G'_b$ with zero error.

Note that if we replace all $*$ and $\dagger$ symbols in Bob's input with 0 or 1, we would get a valid input to to $f \circ G_b$, which we can evaluate using A. Moreover, there is a single special $*$ or $\dagger$ in Bob's input that governs the value of this input to $f \circ G_b$ no matter how we fix the rest of the $*$ and $\dagger$ symbols. Without loss of generality, we assume that if the special symbol is replaced by 0, the function $f \circ G_b$ evaluates to 0, and if it is replaced by 1, it evaluates to 1.

We can now binary search to find this special symbol. There are at most $n2^b$ asterisks and obelisks in Bob's input. We can set the left half to 0 and the right half to 1, and evaluate the resulting input using A. If the answer is 0, the special symbol is on the left half; otherwise, it is on the right half. We can proceed to binary search in this way, until we have zoomed in on one gadget that must contain the special symbol. This requires narrowing down the search space from n possible gadgets to 1, which requires $\log n$ rounds. Each round requires a call to A, times a $O(\log\log n)$ factor for error reduction. We can therefore find the right gadget with bounded error, using $O(\mathrm{R}^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)$ bits of communication.

Once we have found the right gadget, we can certify its validity by having Alice send the right index to Bob, using b bits of communication, and Bob can check that it points to an asterisk or obelisk. Since we found a certificate with constant probability, we can use Lemma 2.5 to turn this into a zero-error algorithm. Thus

$$\mathrm{R}_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b) = O(b + \mathrm{R}^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)\,. \tag{6.2}$$

Since $b = O(\log n)$, we obtain the desired result $\mathrm{R}_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b) = O(\mathrm{R}^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)$. □

Equipped with this lemma we can prove the connection between lifting theorems (Theorem 1.2), stated more precisely as follows.

Theorem 1.2 (Precise version)**.** *Suppose that for all partial Boolean functions f on n bits, we have*

$$\mathrm{R}_0^{\mathrm{cc}}(f \circ G_b) = \Omega(\mathrm{R}_0(f)/\operatorname{polylog} n) \tag{6.3}$$

with $b = O(\log n)$. Then for all partial functions Boolean functions, we also have

$$\mathrm{R}^{\mathrm{cc}}(f \circ G_{2b}) = \Omega(\mathrm{R}(f)/\operatorname{polylog} n)\,. \tag{6.4}$$

The $\operatorname{polylog} n$ *loss in the* R^{cc} *result is only* $\log n \log\log^2 n$ *worse than the loss in the* $\mathrm{R}_0^{\mathrm{cc}}$ *hypothesis.*

Proof. First we show that for any function f and positive integer c,

$$\mathrm{R}^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{\mathrm{R}^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})}{c \log c}\right). \tag{6.5}$$

To see this, note that we can solve $f_{\mathrm{ind}}^{\oplus c} \circ G_{2b}$ by solving the c copies of $f \circ G_{2b}$ and then examining the appropriate cell of the array. This uses $c\,\mathrm{R}^{\mathrm{cc}}(f \circ G_{2b})$ bits of communication, times $O(\log c)$ since we must amplify the randomized protocol to an error of $O(1/c)$.

Next, using (6.5) and Lemma 6.2 on $\mathrm{R}^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})$, we get

$$\mathrm{R}^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{\mathrm{R}^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})}{c \log c}\right) = \Omega\left(\frac{\mathrm{R}_0^{\mathrm{cc}}((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}} \circ G'_{2b})}{c \log c \log n \log\log n}\right). \tag{6.6}$$

From here we want to use the assumed lifting theorem for R_0. However, there is a technicality: the gadget G'_{2b} is not the standard index gadget, and the function $(f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}$ does not have Boolean alphabet. To remedy this, we use two bits to represent each of the symbols $\{0, 1, *, \dagger\}$. Using this representation, we define a new function $(f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}}$ on twice as many bits.

We now compare $(f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}} \circ G_b$ to $(f_{\text{ind}}^{\oplus c})_{\text{usab}} \circ G'_{2b}$. Note that the former uses two pointers of size b to index two bits, while the latter uses one pointer of size $2b$ to index one symbol in $\{0,1,*,\dagger\}$ (which is equivalent to two bits). It's not hard to see that the former function is equivalent to the latter function restricted to a promise. This means the communication complexity of the former is smaller, and hence

$$\mathrm{R}_0^{\mathrm{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}} \circ G'_{2b}) = \Omega(\mathrm{R}_0^{\mathrm{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}} \circ G_b)). \tag{6.7}$$

We are now ready to use the assumed lifting theorem for R_0. To be more precise, let's suppose a lifting result that states $\mathrm{R}_0^{\mathrm{cc}}(f \circ G_b) = \Omega(\mathrm{R}_0(f)/\ell(n))$ for some function $\ell(n)$. Thus

$$\mathrm{R}_0^{\mathrm{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}} \circ G_b) = \Omega(\mathrm{R}_0((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}})/\ell(n)). \tag{6.8}$$

We note that

$$\mathrm{R}_0((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}}) = \Omega(\mathrm{R}_0((f_{\text{ind}}^{\oplus c})_{\text{usab}})) = \Omega(\mathrm{RS}_{\mathrm{u}}(f_{\text{ind}}^{\oplus c})). \tag{6.9}$$

Setting $c = 1.1 \log \mathrm{R}(f)$, we have $\mathrm{RS}_{\mathrm{u}}(f_{\text{ind}}^{\oplus c}) = \Omega(c\,\mathrm{R}(f))$ by Lemma 5.3. Combining this with (6.7), (6.8), and (6.9), we get

$$\mathrm{R}_0^{\mathrm{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}} \circ G'_{2b}) = \Omega(c\,\mathrm{R}(f)/\ell(n)). \tag{6.10}$$

Combining this with (6.6) yields

$$\mathrm{R}^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{c\,\mathrm{R}(f)}{\ell(n) c \log c \log n \log\log n}\right) = \Omega\left(\frac{\mathrm{R}(f)}{\ell(n) \log n \log\log^2 n}\right). \tag{6.11}$$

This gives the desired lifting theorem for bounded-error randomized communication with $\operatorname{polylog} n$ loss that is at most $\log n \log\log^2 n$ worse than the loss in the assumed $\mathrm{R}_0^{\mathrm{cc}}$ lifting theorem. □

7 Comparison with other lower bound methods

In this section we compare $\mathrm{RS}(f)$ with other lower bound techniques for bounded-error randomized query complexity. Figure 1 shows the two most powerful lower bound techniques for $\mathrm{R}(f)$, the partition bound ($\mathrm{prt}(f)$) and quantum query complexity ($\mathrm{Q}(f)$), which subsume all other general lower bound techniques. The partition bound and quantum query complexity are incomparable, since there are functions for which the partition bound is larger, e. g., the OR function, and functions for which quantum query complexity is larger [3]. Another common lower bound measure, approximate polynomial degree ($\widetilde{\deg}$) is smaller than both.

Randomized sabotage complexity (RS) can be much larger than the partition bound and quantum query complexity as we now show. We also show that randomized sabotage complexity is always as large as randomized certificate complexity (RC), which itself is larger than block sensitivity, another common lower bound technique. Lastly, we also show that $\mathrm{R}_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$, showing that RS is a quadratically tight lower bound, even for zero-error randomized query complexity.

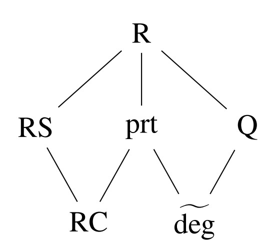

Figure 1: Lower bounds on $\mathrm{R}(f)$.

7.1 Partition bound and quantum query complexity

We start by showing the superiority of randomized sabotage complexity against the two best lower bounds for $\mathrm{R}(f)$. Informally, what we show is that any separation between $\mathrm{R}(f)$ and a lower bound measure like $\mathrm{Q}(f)$, $\mathrm{prt}(f)$, or $\widetilde{\deg}(f)$ readily gives a similar separation between $\mathrm{RS}(f)$ and the same measure.

Theorem 7.1. *There exist total functions f and g such that $\mathrm{RS}(f) \geq \mathrm{prt}(f)^{2-o(1)}$ and $\mathrm{RS}(g) = \widetilde{\Omega}(\mathrm{Q}(g)^{2.5})$. There also exists a total function h with $\mathrm{RS}(h) \geq \widetilde{\deg}(h)^{4-o(1)}$.*

Proof. These separations were shown with $\mathrm{R}(f)$ in place of $\mathrm{RS}(f)$ in [2] and [3]. To get a lower bound on RS, we can simply compose IND with these functions and apply Lemma 5.3. This increases RS to be the same as R (up to logarithmic factors), but it does not increase prt, $\widetilde{\deg}$, or Q more than logarithmically, so the desired separations follow. □

As it turns out, we didn't even need to compose IND with these functions. It suffices to observe that they all use the cheat sheet construction, and that an argument similar to the proof of Lemma 5.3 implies that $\mathrm{RS}(f_{\mathrm{CS}}) = \widetilde{\Omega}(\mathrm{R}(f))$ for all f (where f_{CS} denotes the cheat sheet version of f, as defined in [2]). In particular, cheat sheets can never be used to separate RS from R (by more than logarithmic factors).

7.2 Randomized certificate complexity

Finally, we also show that randomized sabotage complexity upper bounds randomized certificate complexity. To show this, we first define randomized certificate complexity.

Given a string x, a block is a set of bits of x (that is, a subset of $\{1,2,\ldots,n\}$). If B is a block and x is a string, we denote by x^B the string given by flipping the bits specified by B in the string x. If x and x^B are both in the domain of a (possibly partial) function $f : \{0,1\}^n \to \{0,1\}$ and $f(x) \neq f(x^B)$, we say that B is a sensitive block for x with respect to f.

For a string x in the domain f, the maximum number of disjoint sensitive blocks of x is called the block sensitivity of x, denoted by $\mathrm{bs}_x(f)$. The maximum of $\mathrm{bs}_x(f)$ over all x in the domain of f is the block sensitivity of f, denoted by $\mathrm{bs}(f)$.

A fractionally disjoint set of sensitive blocks of x is an assignment of non-negative weights to the sensitive blocks of x such that for all $i \in \{1,2,\ldots,n\}$, the sum of the weights of blocks containing i is at most 1. The maximum total weight of any fractionally disjoint set of sensitive blocks is called the fractional block sensitivity of x. This is also sometimes called the randomized certificate complexity of x, and is denoted by $\mathrm{RC}_x(f)$ [1, 24, 10]. The maximum of this over all x in the domain of f is $\mathrm{RC}(f)$ the randomized certificate complexity of f.

Aaronson [1] observed that $\mathrm{bs}_x(f) \leq \mathrm{RC}_x(f) \leq \mathrm{C}_x(f)$. We therefore have

$$\mathrm{bs}(f) \leq \mathrm{RC}(f) \leq \mathrm{C}(f) \leq \mathrm{R}_0(f) \leq \mathrm{D}(f). \tag{7.1}$$

The measure $\mathrm{RC}(f)$ is also a lower bound for $\mathrm{R}(f)$; indeed, from arguments in [1] it follows that $\mathrm{R}_\varepsilon(f) \geq \mathrm{RC}(f)/(1-2\varepsilon)$, so $\mathrm{R}(f) \geq \mathrm{RC}(f)/3$.

Theorem 7.2. *Let $f : \{0,1\}^n \to \{0,1\}$ be a partial function. Then $\mathrm{RS}(f) \geq \mathrm{RC}(f)/4$.*

Proof. Let us fix x to be any input that maximizes $\mathrm{RC}_x(f)$ (i. e., $\mathrm{RC}(f) = \mathrm{RC}_x(f)$). We will show that any zero-error randomized algorithm that solves the sabotage problem for f makes at least $\mathrm{RC}_x(f)/4$ queries, and hence $\mathrm{RS}(f) \geq \mathrm{RC}_x(f)/4 = \mathrm{RC}(f)/4$.

For our fixed choice of x, let $B_1, B_2, \ldots B_m$ be all the (not necessarily disjoint) sensitive blocks of x. For each $i \in \{1, 2, \ldots, m\}$, let y_i be the sabotaged input formed by replacing block B_i in x with $*$ entries. Note that since each B_i is a sensitive block, each y_i is a sabotaged input (i. e., it is consistent with a 0-input and a 1-input). Hence the optimal zero-error randomized algorithm for the sabotage problem for f when run on one of the inputs in $Y = \{y_1, y_2, \ldots, y_m\}$ will find a $*$ in y_i with at most $\mathrm{RS}(f)$ expected queries.

We now have a randomized algorithm which when run on an input in Y will find a $*$ with at most $\mathrm{RS}(f)$ queries in expectation. By Markov's inequality (Lemma 2.1), we know that this algorithm when run on an input in Y finds a $*$ with probability at least $1/2$ after $\lfloor 2\,\mathrm{RS}(f) \rfloor$ queries.

In general, this is an adaptive algorithm, which decides which bits to query based on the bits it has already queried. The next step is to construct a non-adaptive algorithm (i. e., a probability distribution over input bits) with similar properties. More formally, the probability distribution over input bits should be such that if we make one query to an input in Y using this probability distribution, we would find a $*$ with probability $\Omega(1/\mathrm{RS}(f))$. In particular, this means repeating this non-adaptive algorithm $O(\mathrm{RS}(f))$ times yields an algorithm that finds a $*$ in an input from Y with high probability.

We now use reasoning from [1] to construct the non-adaptive algorithm. For each t between 1 and $T = \lfloor 2\,\mathrm{RS}(f) \rfloor$, let $p_t(y)$ be the probability that the adaptive algorithm when run on input $y \in Y$ finds a $*$ on query t, conditioned on the previous queries not finding a $*$. Then for any $y \in Y$ we have

$$p_1(y) + p_2(y) + \cdots + p_T(y) \geq 1/2\,. \tag{7.2}$$

We now construct a non-adaptive algorithm that finds a $*$ on input y with probability

$$\frac{1}{T} \cdot \sum_{i=1}^{T} p_i(y) \geq \frac{1}{2T}\,. \tag{7.3}$$

Our non-adaptive algorithm first picks a $t \in [T]$ uniformly at random and simulates query t of the adaptive algorithm on input y assuming that the previous queries have not found a $*$. This query can be simulated because we know that previous queries have not found a $*$ and hence the answers to these queries are consistent with x, which is known to us. This non-adaptive algorithm when run on y finds a $*$ with probability equal to the average of the $p_t(y)$, since for a given $t \in [T]$, the probability that it finds a $*$ in y is $p_t(y)$. Hence the non-adaptive algorithm finds a $*$ in any $y \in Y$ with probability

$$\frac{1}{T} \cdot \sum_{i=1}^{T} p_i(y) \geq \frac{1}{2T} \geq \frac{1}{4\,\mathrm{RS}(f)}\,. \tag{7.4}$$

Note that this non-adaptive algorithm does not depend on y, but merely on x, which is fixed.

Let the probability distribution over inputs bits obtained from this non-adaptive algorithm be $(q_1, q_2, \ldots, q_n)$ (i. e., the algorithm queries bit i with probability q_i and $\sum_{i=1}^{n} q_i = 1$). Now for a sensitive block B_j, if y_j is the corresponding input in Y, we know that the algorithm when run on y_j finds a $*$ with probability at least $1/(4\,\mathrm{RS}(f))$, hence it must query inside B_j with probability at least $1/(4\,\mathrm{RS}(f))$, which gives $\sum_{i \in B_j} q_i \geq 1/(4\,\mathrm{RS}(f))$.

For each sensitive block B_j, let w_j be the weight of B_j under the maximum fractional set of disjoint blocks of x. Then $\sum_{j=1}^{m} w_j = \mathrm{RC}_x(f) = \mathrm{RC}(f)$ and for each bit i, we have $\sum_{j:i\in B_j} w_j \leq 1$. We then have

$$\frac{\mathrm{RC}(f)}{4\,\mathrm{RS}(f)} = \sum_{j=1}^{m} w_j \cdot \frac{1}{4\,\mathrm{RS}(f)} \leq \sum_{j=1}^{m} w_j \sum_{i\in B_j} q_i = \sum_{i=1}^{n} q_i \sum_{j:i\in B_j} w_j \leq \sum_{i=1}^{n} q_i \cdot 1 \leq 1\,. \tag{7.5}$$

Hence $\mathrm{RS}(f) \geq \mathrm{RC}(f)/4$. □

7.3 Zero-error randomized query complexity

Theorem 7.3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a total function. Then $\mathrm{R}_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$ or alternately, $\mathrm{RS}(f) = \Omega(\sqrt{\mathrm{R}_0(f)/\log \mathrm{R}_0(f)})$.*

Proof. Let A be the $\mathrm{RS}(f)$ algorithm. The idea will be to run A on an input to x for long enough that we can ensure it queries a bit in every sensitive block of x; this will mean A found a certificate for x. That will allow us to turn the algorithm into a zero-error algorithm for f.

Let x be any input, and let b be a sensitive block of x. If we replace the bits of x specified by b with stars, then we can find a $*$ with probability $1/2$ by running A for $2\,\mathrm{RS}(f)$ queries by Lemma 2.1. This means that if we run A on x for $2\,\mathrm{RS}(f)$ queries, it has at least $1/2$ probability of querying a bit in any given sensitive block of x. If we repeat this k times, we get a $2k\,\mathrm{RS}(f)$ query algorithm that queries a bit in any given sensitive block of x with probability at least $1-2^{-k}$.

Now, by [18], the number of minimal sensitive blocks in x is at most $\mathrm{RC}(f)^{\mathrm{bs}(f)}$ for a total function f. Our probability of querying a bit in all of these sensitive blocks is at least $1-2^{-k}\,\mathrm{RC}(f)^{\mathrm{bs}(f)}$ by the union bound. When $k \geq 1+\mathrm{bs}(f)\log_2 \mathrm{RC}(f)$, this is at least $1/2$. Since a bit from every sensitive block is a certificate, by Lemma 2.5, we can turn this into a zero-error randomized algorithm with expected query complexity at most $4(1+\mathrm{bs}(f)\log_2 \mathrm{RC}(f))\,\mathrm{RS}(f)$, which gives $\mathrm{R}_0(f) = O(\mathrm{RS}(f)\,\mathrm{bs}(f)\log \mathrm{RC}(f))$. Since $\mathrm{bs}(f) \leq \mathrm{RC}(f) = O(\mathrm{RS}(f))$ by Theorem 7.2, we have $\mathrm{R}_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$, or $\mathrm{RS}(f) = \Omega(\sqrt{\mathrm{R}_0(f)/\log \mathrm{R}_0(f)})$. □

8 Deterministic sabotage complexity

Finally we look at the deterministic analogue of randomized sabotage complexity. It turns out that deterministic sabotage complexity (as defined in Definition 3.2) is exactly the same as deterministic query complexity for all (partial) functions. Since we already know perfect composition and direct sum results for deterministic query complexity, it is unclear if deterministic sabotage complexity has any applications.

Theorem 8.1. *Let $f : \{0,1\}^n \to \{0,1\}$ be a partial function. Then $\mathrm{DS}(f) = \mathrm{D}(f)$.*

Proof. For any function $\mathrm{DS}(f) \leq \mathrm{D}(f)$ since a deterministic algorithm that correctly computes f must find a $*$ or $\dagger$ when run on a sabotaged input, otherwise its output is independent of how the sabotaged bits are filled in.

To show the other direction, let $\mathrm{D}(f) = k$. This means for every $k-1$ query algorithm, there are two inputs x and y with $f(x) \neq f(y)$, such that they have the same answers to the queries made by the algorithm. If this is not the case then this algorithm computes $f(x)$, contradicting the fact that $\mathrm{D}(f) = k$.

Thus if there is a deterministic algorithm for f_{sab} that makes $k-1$ queries, there exist two inputs x and y with $f(x) \neq f(y)$ that have the same answers to the queries made by the algorithm. If we fill in the rest of the inputs bits with either asterisks or obelisks, it is clear that this is a sabotaged input (since it can be completed to either x or y), but the purported algorithm for f_{sab} cannot distinguish them. Hence $\mathrm{D}(f_{\mathrm{sab}}) \geq k$, which means $\mathrm{DS}(f) \geq \mathrm{D}(f)$. □

Acknowledgements

We thank Mika Göös for finding an error in an earlier proof of Theorem 4.4. We also thank the anonymous referees of Theory of Computing for their comments.

A Properties of randomized algorithms

We now provide proofs of the properties described in Section 2, which we restate for convenience.

Lemma A.1 (Markov's Inequality)**.** *Let A be a randomized algorithm that makes T queries in expectation (over its internal randomness). Then for any $\delta \in (0,1)$, the algorithm A terminates within $\lfloor T/\delta \rfloor$ queries with probability at least $1-\delta$.*

Proof. If A does not terminate within $\lfloor T/\delta \rfloor$ queries, it must use at least $\lfloor T/\delta \rfloor + 1$ queries. Let's say this happens with probability p. Then the expected number of queries used by A is at least $p(\lfloor T/\delta \rfloor + 1)$ (using the fact that the number of queries used is always non-negative). We then get $T \geq p(\lfloor T/\delta \rfloor + 1) > pT/\delta$, or $p < \delta$. Thus A terminates within T/δ queries with probability greater than $1-\delta$. □

Lemma A.2 (Amplification)**.** *If f is a function with Boolean output and A is a randomized algorithm for f with error $\varepsilon < 1/2$, repeating A several times and taking the majority vote of the outcomes decreases the error. To reach error $\varepsilon' > 0$, it suffices to repeat the algorithm $2\ln(1/\varepsilon')/(1-2\varepsilon)^2$ times.*

Proof. Let's repeat A an odd number of times, say $2k+1$. The error probability of A', the algorithm that takes the majority vote of these runs, is

$$\sum_{i=0}^{k} \binom{2k+1}{i} \varepsilon^{2k+1-i}(1-\varepsilon)^i \leq \varepsilon^{2k+1-k}(1-\varepsilon)^k \sum_{i=0}^{k} \binom{2k+1}{i}, \tag{A.1}$$

which is at most

$$\varepsilon^{k+1}(1-\varepsilon)^k(2^{2k+1}/2) = \varepsilon^{k+1}(1-\varepsilon)^k 4^k = \varepsilon(4\varepsilon(1-\varepsilon))^k = \varepsilon(1-(1-2\varepsilon)^2)^k. \tag{A.2}$$

It suffices to choose k large enough so that $\varepsilon(1-(1-2\varepsilon)^2)^k \leq \varepsilon'$, or $\ln\varepsilon + k\ln(1-(1-2\varepsilon)^2) \leq \ln\varepsilon'$. Using the inequality $\ln(1-x) \leq -x$, it suffices to choose k so that $k(1-2\varepsilon)^2 \geq \ln(1/\varepsilon') - \ln(1/\varepsilon)$, or

$$k \geq \frac{\ln(1/\varepsilon') - \ln(1/\varepsilon)}{(1-2\varepsilon)^2}. \tag{A.3}$$

In particular, we can choose

$$k = \left\lceil \frac{\ln(1/\varepsilon') - \ln(1/\varepsilon)}{(1-2\varepsilon)^2} \right\rceil \le \frac{\ln(1/\varepsilon')}{(1-2\varepsilon)^2} + 1 - \frac{\ln(1/\varepsilon)}{(1-2\varepsilon)^2}. \tag{A.4}$$

It is not hard to check that $3(1-2\varepsilon)^2 \le 2\ln(1/\varepsilon)$ for all $\varepsilon \in (0,1/2)$, so we can choose k to be at most

$$\frac{\ln(1/\varepsilon')}{(1-2\varepsilon)^2} - 1/2. \tag{A.5}$$

This means $2k+1$ is at most

$$\frac{2\ln(1/\varepsilon')}{(1-2\varepsilon)^2}, \tag{A.6}$$

as desired. □

Lemma A.3. *Let f be a partial function, $\delta > 0$, and $\varepsilon \in [0,1/2)$. Then we have*

$$\mathrm{R}_{\varepsilon+\delta}(f) \le \frac{1-2\varepsilon}{2\delta}\overline{\mathrm{R}}_{\varepsilon}(f) \le \frac{1}{2\delta}\overline{\mathrm{R}}_{\varepsilon}(f).$$

Proof. Let A be the $\overline{\mathrm{R}}_{\varepsilon}(f)$ algorithm. Let B be the algorithm that runs A for $\lfloor \overline{\mathrm{R}}_{\varepsilon}(f)/\alpha \rfloor$ queries, and if A doesn't terminate, outputs 0 with probability $1/2$ and 1 with probability $1/2$. Then by Lemma 2.1, the error probability of B is at most $\alpha/2 + (1-\alpha)\varepsilon$. If we let $\alpha = 2\delta/(1-2\varepsilon)$, then the error probability of B is at most

$$\frac{\delta}{1-2\varepsilon} + \frac{(1-2\varepsilon-2\delta)\varepsilon}{1-2\varepsilon} = \varepsilon + \delta, \tag{A.7}$$

as desired. The number of queries made by B is at most

$$\lfloor \overline{\mathrm{R}}_{\varepsilon}(f)/\alpha \rfloor \le \frac{1-2\varepsilon}{2\delta}\overline{\mathrm{R}}_{\varepsilon}(f) \le \frac{1}{2\delta}\overline{\mathrm{R}}_{\varepsilon}(f). \tag{A.8}$$

□

Lemma A.4. *If f is a partial function, then for all $\varepsilon \in (0,1/2)$, we have*

$$\mathrm{R}_{\varepsilon}(f) \le 14\frac{\ln(1/\varepsilon)}{(1-2\varepsilon)^2}\overline{\mathrm{R}}_{\varepsilon}(f).$$

When $\varepsilon = 1/3$, we can improve this to $\mathrm{R}(f) \le 10\overline{\mathrm{R}}(f)$.

Proof. Repeating an algorithm with error $1/3$ three times decreases its error to $7/27$, so in particular $\overline{\mathrm{R}}_{7/27}(f) \le 3\overline{\mathrm{R}}(f)$. Then using Lemma 2.3 with $\varepsilon + \delta = 1/3$ and $\varepsilon = 7/27$, we get

$$\mathrm{R}(f) \le \frac{1-14/27}{2(1/3-7/27)}\overline{\mathrm{R}}_{7/27}(f) = \frac{13}{4}\overline{\mathrm{R}}_{7/27}(f) \le \frac{39}{4}\overline{\mathrm{R}}(f) \le 10\overline{\mathrm{R}}(f). \tag{A.9}$$

To deal with arbitrary ε, we need to use Lemma 2.2. It gives us

$$\mathrm{R}_{\varepsilon'}(f) \le \frac{2\ln(1/\varepsilon')}{(1-2\varepsilon)^2}\mathrm{R}_{\varepsilon}(f). \tag{A.10}$$

When combined with Lemma 2.3, this gives

$$\mathrm{R}_{\varepsilon'}(f) \le \frac{1-2\varepsilon'}{\varepsilon-\varepsilon'}\frac{\ln(1/\varepsilon')}{(1-2\varepsilon)^2}\overline{\mathrm{R}}_{\varepsilon'}(f)\,. \tag{A.11}$$

Setting $\varepsilon = (1+4\varepsilon')/6$ (which is greater than ε' if $\varepsilon' < 1/2$) gives

$$\mathrm{R}_{\varepsilon'}(f) \le \frac{27\ln(1/\varepsilon')}{2(1-2\varepsilon')^2}\overline{\mathrm{R}}_{\varepsilon'}(f) \le 14\frac{\ln(1/\varepsilon')}{(1-2\varepsilon')^2}\overline{\mathrm{R}}_{\varepsilon'}(f)\,, \tag{A.12}$$

which gives the desired result (after exchanging ε and ε'). □

Lemma A.5. *Let A be a randomized algorithm that uses T queries in expectation and finds a certificate with probability $1-\varepsilon$. Then repeating A when it fails to find a certificate turns it into an algorithm that always finds a certificate (i. e., a zero-error algorithm) that uses at most $T/(1-\varepsilon)$ queries in expectation.*

Proof. Let A' be the algorithm that runs A, checks if it found a certificate, and repeats if it didn't. Let N_1 be the random variable for the number of queries used by A'. We know that the maximum number of queries A' ever uses is the input size; it follows that $\mathbb{E}(N_1)$ converges and is at most the input size.

Let M_1 be the random variable for the number of queries used by A in the first iteration. Let S_1 be the Bernoulli random variable for the event that A fails to find a certificate. Then $\mathbb{E}(M_1) = T$ and $\mathbb{E}(S_1) = \varepsilon$. Let N_2 be the random variable for the number of queries used by A' starting from the second iteration (conditional on the first iteration failing). Then

$$N_1 = M_1 + S_1 N_2\,, \tag{A.13}$$

so by linearity of expectation and independence,

$$\mathbb{E}(N_1) = \mathbb{E}(M_1) + \mathbb{E}(S_1)\mathbb{E}(N_2) = T + \varepsilon\mathbb{E}(N_2) \le T + \varepsilon\mathbb{E}(N_1)\,. \tag{A.14}$$

This implies

$$\mathbb{E}(N_1) \le T/(1-\varepsilon)\,, \tag{A.15}$$

as desired. □

Lemma A.6. *Let f be a partial function. Let A be a randomized algorithm that solves f using at most T expected queries and with error at most ε. For $x, y \in \mathrm{Dom}(f)$ if $f(x) \ne f(y)$ then when A is run on x, it must query an entry on which x differs from y with probability at least $1-2\varepsilon$.*

Proof. Let p be the probability that A queries an entry on which x differs from y when it is run on x. Let q be the probability that A outputs an invalid output for x given that it doesn't query a difference from y. Let r be the probability that A outputs an invalid output for y given that it doesn't query such a difference. Since one of these events always happens, we have $q + r \ge 1$. Note that A errs with probability at least $(1-p)q$ when run on x and at least $(1-p)r$ when run on y. This means that $(1-p)q \le \varepsilon$ and $(1-p)r \le \varepsilon$. Summing these gives $1-p \le (1-p)(q+r) \le 2\varepsilon$, so $p \ge 1-2\varepsilon$, as desired. □

B Minimax theorem for bounded-error algorithms

We need the following version of Yao's minimax theorem for $\overline{\mathrm{R}}_\varepsilon(f)$. The proof is similar to other minimax theorems in the literature, but we include it for completeness.

Theorem B.1. *Let f be a partial function and $\varepsilon \geq 0$. Then there exists a distribution μ over $\mathrm{Dom}(f)$ such that any randomized algorithm A that computes f with error at most ε on all $x \in \mathrm{Dom}(f)$ satisfies $\mathbb{E}_{x\sim\mu}A(x) \geq \overline{\mathrm{R}}_\varepsilon(f)$, where $A(x)$ is the expected number of queries made by A on x.*

We note that Theorem B.1 talks only about algorithms that successfully compute f (with error ε) on all inputs, not just those sampled from μ. An alternative minimax theorem where the error is with respect to the distribution μ can be found in [25], although it loses constant factors.

Proof. We think of a randomized algorithm as a probability vector over deterministic algorithms; thus randomized algorithms lie in $\mathbb{R}^N$, where N is the number of deterministic decision trees. In fact, the set S of randomized algorithms forms a simplex, which is a closed and bounded set.

Let $\mathrm{err}_{f,x}(A) := \Pr[A(x) \neq f(x)]$ be the probability of error of the randomized A when run on x. Then it is not hard to see that $\mathrm{err}_{f,x}(A)$ is a continuous function of A. Define $\mathrm{err}_f(A) := \max_x \mathrm{err}_{f,x}(A)$. Then $\mathrm{err}_f(A)$ is also the maximum of a finite number of continuous functions, so it is continuous.

Next consider the set of algorithms $S_\varepsilon := \{A \in S : \mathrm{err}_f(A) \leq \varepsilon\}$. Since $\mathrm{err}_f(A)$ is a continuous function and S is closed and bounded, it follows that S_ε is closed and bounded, and hence compact. It is also easy to check that S_ε is convex. Let P be the set of probability distributions over $\mathrm{Dom}(f)$. Then P is also compact and convex. Finally, consider the function $\alpha(A,\mu) := \mathbb{E}_{x\sim\mu}\mathbb{E}_{D\sim A}D(x)$ that accepts a randomized algorithm and a distribution as input, and returns the expected number of queries the algorithm makes on that distribution. It is not hard to see that α is a continuous function in both variables. In fact, α is linear in both variables by the linearity of expectation.

Since S_ε and P are compact and convex subsets of the finite-dimensional spaces $\mathbb{R}^N$ and $\mathbb{R}^{\mathrm{Dom}(f)}$, respectively, and the objective function $\alpha(\cdot,\cdot)$ is linear, we can apply Sion's minimax theorem (see [23] or [26, Theorem 1.12]) to get

$$\max_{\mu\in P}\min_{A\in S_\varepsilon}\alpha(A,\mu) = \min_{A\in S_\varepsilon}\max_{\mu\in P}\alpha(A,\mu)\,. \tag{B.1}$$

The right hand side is simply the worst-case expected query complexity of any algorithm computing f with error at most ε, which is $\overline{\mathrm{R}}_\varepsilon(f)$ by definition. The left hand side gives us a distribution μ such that for any algorithm A that makes error at most ε on all $x \in \mathrm{Dom}(f)$, the expected number of queries A makes on μ is at least $\overline{\mathrm{R}}_\varepsilon(f)$. □

References

[1] SCOTT AARONSON: Quantum certificate complexity. *J. Comput. System Sci.*, 74(3):313–322, 2008. Preliminary version in CCC'03. [doi:10.1016/j.jcss.2007.06.020, arXiv:quant-ph/0210020] 18, 19

[2] SCOTT AARONSON, SHALEV BEN-DAVID, AND ROBIN KOTHARI: Separations in query complexity using cheat sheets. In *Proc. 48th STOC*, pp. 863–876. ACM Press, 2016. [doi:10.1145/2897518.2897644, arXiv:1511.01937] 3, 4, 18

[3] ANDRIS AMBAINIS, MARTINS KOKAINIS, AND ROBIN KOTHARI: Nearly optimal separations between communication (or query) complexity and partitions. In *Proc. 31st IEEE Conf. on Computational Complexity (CCC'16)*, pp. 4:1–4:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. [doi:10.4230/LIPIcs.CCC.2016.4, arXiv:1512.01210] 4, 17, 18

[4] 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 *Proc. 57st FOCS*, pp. 555–564. IEEE Comp. Soc. Press, 2016. [doi:10.1109/FOCS.2016.66, arXiv:1605.01142] 4

[5] BOAZ BARAK, MARK BRAVERMAN, XI CHEN, AND ANUP RAO: How to compress interactive communication. *SIAM J. Computing*, 42(3):1327–1363, 2013. Preliminary version in STOC'10. [doi:10.1137/100811969] 2

[6] SHALEV BEN-DAVID AND ROBIN KOTHARI: Randomized query complexity of sabotaged and composed functions. In *Proc. 43rd Internat. Colloq. on Automata, Languages and Programming (ICALP'16)*, pp. 60:1–60:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. [doi:10.4230/LIPIcs.ICALP.2016.60, arXiv:1605.09071] 1

[7] HARRY BUHRMAN AND RONALD DE WOLF: Complexity measures and decision tree complexity: A survey. *Theoret. Computer Sci.*, 288(1):21–43, 2002. [doi:10.1016/S0304-3975(01)00144-X] 6

[8] ANDREW DRUCKER: Improved direct product theorems for randomized query complexity. *Comput. Complexity*, 21(2):197–244, 2012. Preliminary version in CCC'11. [doi:10.1007/s00037-012-0043-7, arXiv:1005.0644] 13

[9] TOMÀS FEDER, EYAL KUSHILEVITZ, MONI NAOR, AND NOAM NISAN: Amortized communication complexity. *SIAM J. Computing*, 24(4):736–750, 1995. [doi:10.1137/S0097539792235864] 2

[10] JUSTIN GILMER, MICHAEL SAKS, AND SRIKANTH SRINIVASAN: Composition limits and separating examples for some Boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016. Preliminary version in CCC'13. [doi:10.1007/s00493-014-3189-x, arXiv:1306.0630] 18

[11] MIKA GÖÖS, TONIANN PITASSI, AND THOMAS WATSON: Deterministic communication vs. partition number. In *Proc. 56st FOCS*, pp. 1077–1088. IEEE Comp. Soc. Press, 2015. [doi:10.1109/FOCS.2015.70] 4, 15

[12] PETER HØYER, TROY LEE, AND ROBERT ŠPALEK: Negative weights make adversaries stronger. In *Proc. 39th STOC*, pp. 526–535. ACM Press, 2007. [doi:10.1145/1250790.1250867, arXiv:quant-ph/0611054] 3

[13] RAHUL JAIN AND HARTMUT KLAUCK: The partition bound for classical communication complexity and query complexity. In *Proc. 25th IEEE Conf. on Computational Complexity (CCC'10)*, pp. 247–258. IEEE Comp. Soc. Press, 2010. [doi:10.1109/CCC.2010.31, arXiv:0910.4266] 4

[14] RAHUL JAIN, HARTMUT KLAUCK, AND MIKLOS SANTHA: Optimal direct sum results for deterministic and randomized decision tree complexity. *Inform. Process. Lett.*, 110(20):893–897, 2010. [doi:10.1016/j.ipl.2010.07.020, arXiv:1004.0105] 2, 10

[15] RAHUL JAIN, TROY LEE, AND NISHEETH K. VISHNOI: A quadratically tight partition bound for classical communication complexity and query complexity, 2014. Available at author's webpage. [arXiv:1401.4512] 4

[16] MAURICIO KARCHMER, RAN RAZ, AND AVI WIGDERSON: Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Comput. Complexity*, 5(3-4):191–204, 1995. Preliminary version in SCT'91. [doi:10.1007/BF01206317] 2

[17] SHELBY KIMMEL: Quantum adversary (upper) bound. *Chicago J. Theoret. Computer Sci.*, (4), 2013. [doi:10.4086/cjtcs.2013.004] 2

[18] RAGHAV KULKARNI AND AVISHAY TAL: On fractional block sensitivity. *Chicago J. Theoret. Computer Sci.*, 2016(8), 2016. [doi:10.4086/cjtcs.2016.008] 20

[19] TROY LEE, RAJAT MITTAL, BEN W. REICHARDT, ROBERT ŠPALEK, AND MARIO SZEGEDY: Quantum query complexity of state conversion. In *Proc. 52nd FOCS*, pp. 344–353. IEEE Comp. Soc. Press, 2011. [doi:10.1109/FOCS.2011.75, arXiv:1011.3020] 2

[20] ASHLEY MONTANARO: A composition theorem for decision tree complexity. *Chicago J. Theoret. Computer Sci.*, 2014(6), 2014. [doi:10.4086/cjtcs.2014.006, arXiv:1302.4207] 2

[21] DENIS PANKRATOV: Direct sum questions in classical communication complexity. Master's thesis, University of Chicago, 2012. Available at advisor's webpage. 2

[22] BEN W. REICHARDT: Reflections for quantum query algorithms. In *Proc. 22nd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'11)*, pp. 560–569. ACM Press, 2011. [doi:10.1137/1.9781611973082.44] 2

[23] MAURICE SION: On general minimax theorems. *Pacific J. Math.*, 8(1):171–176, 1958. [doi:10.2140/pjm.1958.8.171] 24

[24] AVISHAY TAL: Properties and applications of Boolean function composition. In *Proc. 4th Innovations in Theoret. Computer Sci. Conf. (ITCS'13)*, pp. 441–454. ACM Press, 2013. [doi:10.1145/2422436.2422485] 2, 18

[25] NIKOLAI K. VERESHCHAGIN: Randomized Boolean decision trees: Several remarks. *Theoret. Computer Sci.*, 207(2):329–342, 1998. [doi:10.1016/S0304-3975(98)00071-1] 24

[26] JOHN WATROUS: *The Theory of Quantum Information*. Cambridge University Press, 2018. Available at CUP and at author's webpage. 24

[27] ANDREW CHI-CHIH YAO: Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18st FOCS*, pp. 222–227. IEEE Comp. Soc. Press, 1977. [doi:10.1109/SFCS.1977.24] 9

AUTHORS

Shalev Ben-David
Hartree postdoctoral fellow
Joint Center for Quantum Information
and Computer Science
University of Maryland
College Park, MD, USA
shalev@umd.edu

Robin Kothari
Researcher
Quantum Architectures and Computation group (QuArC)
Microsoft Research
Redmond, WA, USA
robin.kothari@microsoft.com
http://www.robinkothari.com

ABOUT THE AUTHORS

SHALEV BEN-DAVID received his Ph. D. in computer science from M.I.T. in 2017, under the advisorship of Scott Aaronson. His research interests include complexity theory and quantum computing. He is currently a Hartree postdoctoral fellow at the University of Maryland, though this paper was written when he was a student at MIT. He is always looking for people to play Hex with.

ROBIN KOTHARI received his Ph. D. in Computer Science from the University of Waterloo in 2014, under the supervision of Andrew Childs and John Watrous. This research was performed while he was a postdoctoral associate at the Center for Theoretical Physics at the Massachusetts Institute of Technology. He is currently a Researcher in the Quantum Architectures and Computation group (QuArC) at Microsoft Research. His research interests include quantum algorithms and complexity theory.