Randomized Query Complexity of Sabotaged and Composed Functions*

Shalev Ben-David[1] and Robin Kothari[2]

1 Massachusetts Institute of Technology, Cambridge, MA, USA
shalev@mit.edu
2 Massachusetts Institute of Technology, Cambridge, MA, USA
rkothari@mit.edu

Abstract

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

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, $R(f \circ g) = \Omega(R(f)\,\mathrm{RS}(g))$. 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 from zero-error randomized query to communication complexity implies a similar result for bounded-error algorithms for all total functions.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases Randomized query complexity, decision tree complexity, composition theorem, partition bound, lifting theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.60

1 Introduction

1.1 Composition theorems

A basic structural question that can be asked in any model of computation is whether there can be savings in complexity 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. A direct sum theorem is known to hold for deterministic and randomized query complexity [9], and two-player refereed games [17], is known to fail for circuit size [16], and remains open for deterministic communication complexity [11].

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 $f \circ g(x_1, \ldots, x_n) = f(g(x_1), \ldots, g(x_n))$, 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

* This work was partially supported by ARO grant number W911NF-12-1-0486.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 60; pp. 60:1–60:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 and quantum query complexity. Proving this for randomized query complexity remains a major open problem. More precisely, if $D(f)$, $R(f)$, and $Q(f)$ denote the deterministic, randomized, and quantum query complexities of f, then we know for all partial Boolean functions f and g, $D(f \circ g) = D(f)D(g)$ [18, 15] and $Q(f \circ g) = \Theta(Q(f)Q(g))$ [14, 12]. (Such theorems often fail for functions with non-Boolean output, and hence we only consider functions with Boolean output in this paper.) In contrast, in the randomized setting we have only the upper bound $R(f \circ g) = O(R(f)R(g) \log R(f))$.

▶ **Open Problem 1.** Does it hold that $R(f \circ g) = \Omega(R(f)R(g))$ for all Boolean 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 $\Theta(\log R(g))$. 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 $k = \Theta(\log R(g))$ in our construction.

▶ **Theorem 1.** *Let f and g be partial Boolean functions and let* $\textsc{Ind}$ *be the index function with* $R(\textsc{Ind}) = \Theta(\log R(g))$*. Then* $R(f \circ \textsc{Ind} \circ g) = \Omega(R(f)R(\textsc{Ind})R(g)) = \Omega(R(f)R(g)\log R(g))$.

Theorem 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 f is the AND function [2, 5] or when g is the AND function [1]. Our composition theorem would suffice for both applications.

We prove Theorem 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 general adversary bound [7] 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 $x_i \in \{0,1\}$. Now imagine a saboteur damages the oracle making 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, and 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. This can be formalized and leads to a lower bound measure for several models of computation, including deterministic, randomized, and quantum query complexity.

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 this problem by imaging there are two saboteurs, one of whom 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 $\mathrm{DS}(f) := D(f_{\text{sab}})$ and in fact, $\mathrm{DS}(f) = D(f)$ as we show in the full version of this paper. We could define the *randomized sabotage complexity of* f as $R(f_{\text{sab}})$, but instead we define it as $\mathrm{RS}(f) := R_0(f_{\text{sab}})$, where R_0 denotes zero-error randomized query complexity, since $R(f_{\text{sab}})$ and $R_0(f_{\text{sab}})$ are equal up to constant factors. Besides lower bounding $R(f)$, $\mathrm{RS}(f)$ has the following desirable properties:

1. (Perfect composition) For all f and g, $\mathrm{RS}(f \circ g) \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$ (Theorem 15)
2. (Composition with R) For all f and g, $R(f \circ g) = \Omega(R(f)\,\mathrm{RS}(g))$ (Theorem 17)
3. (Quadratically tight) For all total f, $R(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$ (Theorem 28)
4. (Superior to $\mathrm{prt}(f)$) There exists a total f with $\mathrm{RS}(f) \geq \mathrm{prt}(f)^{2-o(1)}$ (Theorem 26)

Here $\mathrm{prt}(f)$ denotes the partition bound [8, 10], which subsumes most other lower bound techniques such as approximate polynomial degree and randomized certificate complexity. In fact, we are unaware of any total function f for which $\mathrm{RS}(f) = o(R(f))$, leaving open the intriguing possibility that this lower bound technique is tight.

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 [6] showed that there is a communication problem G with communication complexity $\Theta(\log n)$ such that for any function f on n bits, $D^{\text{cc}}(f \circ G) = \Omega(D(f) \log n)$, where D^{cc} denotes deterministic communication complexity.

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 complexity for total functions. We use R_0^{cc} and R^{cc} to denote zero-error and bounded-error communication complexity respectively.

▶ **Theorem 2.** *Let G be the communication gadget from [6] with $D^{\text{cc}}(G) = \Theta(\log n)$. If it holds that for all n-bit (possibly partial) functions f, $R_0^{\text{cc}}(f \circ G) = \Omega(R_0(f)/\operatorname{polylog} n)$, then it holds that for all n-bit total Boolean functions f, $R^{\text{cc}}(f \circ G) = \Omega(R(f)/\operatorname{polylog} n)$.*

Proving a lifting theorem for bounded-error randomized query complexity remains an important open problem, and would imply super-quadratic separations between randomized and quantum communication complexity [1], and a nearly quadratic separation between randomized communication complexity and partition number [2]. Our result shows that it is sufficient to prove a lifting theorem for zero-error randomized protocols instead.

2 Preliminaries

We now define some basic notions in query complexity. Note that all the functions in this paper have Boolean output. 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 $D(f)$.

A randomized algorithm is a probability distribution over deterministic algorithms. The worst-case cost of a randomized algorithm is the worst-case 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 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 $1-\epsilon$.

We use $R_\epsilon(f)$ to denote the worst-case cost of the best randomized algorithm that computes f with error ϵ. Similarly, we use $\overline{R}_\epsilon$ 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 $\epsilon = 1/3$. Thus $R(f)$ denotes the bounded-error randomized query complexity of f. Finally, we also define zero-error randomized query complexity, which is $\overline{R}_0(f)$, which we also denote by $R_0(f)$ to be consistent with the literature. For precise definitions of these measures as well as the definition of quantum query complexity $Q(f)$, see [3]. We also need two simple properties of randomized algorithms, which we prove in the full version of this paper.

▶ **Lemma 3.** *If A is a randomized algorithm that uses T expected queries and finds a certificate with probability $1-\epsilon$, then repeating A when it fails turns it into a zero-error algorithm that uses at most $T/(1-\epsilon)$ expected queries.*

▶ **Lemma 4.** *Let f be a partial function and A be an ϵ-error randomized algorithm for f that uses at most T expected queries. 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 $1-2\epsilon$.*

3 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 † 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 5.** 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 $q \in P_f^\dagger$.

See Section 1.2 for more discussion and motivation for this definition. Now that we have defined f_{sab}, we can define sabotage complexity for various models.

▶ **Definition 6.** Let f be a partial function. Then $\mathrm{DS}(f) := D(f_{\mathrm{sab}})$ and $\mathrm{RS}(f) := R_0(f_{\mathrm{sab}})$.

We will primarily focus on $\mathrm{RS}(f)$ in this work. To justify defining $\mathrm{RS}(f)$ as $R_0(f_{\mathrm{sab}})$ instead of $R(f_{\mathrm{sab}})$, we now show these definitions are equivalent up to constant factors.

▶ **Theorem 7.** *Let f be a partial function. Then $R_0(f_{\mathrm{sab}}) \geq \overline{R}_\epsilon(f_{\mathrm{sab}}) \geq (1-2\epsilon)R_0(f_{\mathrm{sab}})$.*

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 4 any ϵ-error randomized algorithm that solves f_{sab} must find a position on which x and x' differ with probability at least $1-2\epsilon$. 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 $1-2\epsilon$. Since finding an asterisk or obelisk is a certificate for f_{sab}, by Lemma 3, we get a zero-error algorithm for f_{sab} that uses $\overline{R}_\epsilon(f_{\mathrm{sab}})/(1-2\epsilon)$ expected queries. Thus $R_0(f_{\mathrm{sab}}) \leq \overline{R}_\epsilon(f_{\mathrm{sab}})/(1-2\epsilon)$, as desired. ◀

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

▶ **Theorem 8.** *Let f be an n-bit partial function. Then $R_\epsilon(f) \geq \overline{R}_\epsilon(f) \geq (1-2\epsilon)\,\mathrm{RS}(f)$.*

Proof. Let A be a randomized algorithm for f that uses $\overline{R}_\epsilon(f)$ randomized queries and outputs the correct answer on every input in $\mathrm{Dom}(f)$ with probability at least $1-\epsilon$. 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 $q(1-p) \leq \epsilon$ and $(1-q)(1-p) \leq \epsilon$. Summing them gives $1-p \leq 2\epsilon$, or $p \geq 1-2\epsilon$.

This means A finds a $*$ entry within $\overline{R}_\epsilon(f)$ expected queries with probability at least $1-2\epsilon$. By Lemma 3, we get $\frac{1}{1-2\epsilon}\overline{R}_\epsilon(f) \geq \mathrm{RS}(f)$, or $\overline{R}_\epsilon(f) \geq (1-2\epsilon)\,\mathrm{RS}(f)$. ◀

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

▶ **Definition 9.** Let f be a partial function. We define f_{usab} as the restriction of f_{sab} to U, the set of strings with only one asterisk or obelisk. I.e., f_{usab} has domain $Q_f \cap U$, but is equal to f_{sab} on its domain. We then define $\mathrm{RS}_1(f) := R_0(f_{\mathrm{usab}})$. If $Q_f \cap U$ is empty, we define $\mathrm{RS}_1(f) := 0$.

The measure RS_1 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 is smaller, so $\mathrm{RS}_1(f) \leq \mathrm{RS}(f)$. Another interesting property is the following theorem, which says $\mathrm{RS}_1(f)$ equals $\mathrm{RS}(f)$ for total functions. 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 10.** *If f is a total function, then $\mathrm{RS}_1(f) = \mathrm{RS}(f)$.*

Proof. We showed that $\mathrm{RS}(f) \geq \mathrm{RS}_1(f)$. To show $\mathrm{RS}_1(f) \geq \mathrm{RS}(f)$, we argue that any zero-error algorithm A for f_{usab} also solves f_{sab}. The main observation 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 †, 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.

Now, the algorithm A must find an asterisk or obelisk in any input to f_{usab}. But since each input to f_{sab} can be viewed as an input to f_{usab} with added asterisks and obelisks, the algorithm A also finds an asterisk or obelisk in any input to f_{sab}. Thus $\mathrm{RS}(f) = R_0(f_{\text{sab}}) \leq R_0(f_{\text{usab}}) = \mathrm{RS}_1(f)$. ◀

4 Direct Sum and Composition Theorems

In this section, we establish some composition theorems for RS. 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). We will require Yao's minimax theorem [19]:

▶ **Theorem 11** (Yao). *Let f be a partial function. There is a distribution μ over inputs in $\mathrm{Dom}(f)$ such that all zero-error algorithms for f use at least $R_0(f)$ expected queries on μ.*

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 12.** 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 all $x_i \in \mathrm{Dom}(f)$,

$$f(x_1, x_2, \ldots, x_m) = (f(x_1), f(x_2), \ldots, f(x_m)). \tag{1}$$

We can now prove a direct sum theorem for zero-error randomized query complexity. We prove these results for partial functions, although they also hold for relations.

▶ **Theorem 13** (Direct sum). *For any n-bit partial function f and any positive integer m, we have $R_0(f^{\oplus m}) = mR_0(f)$. Moreover, if μ is the hard distribution for f given by Theorem 11, then $\mu^{\otimes m}$ is a hard distribution for $f^{\oplus m}$.*

Proof. The upper bound follows from running the $R_0(f)$ algorithm on each of the m inputs to f. By linearity of expectation, this solves all m inputs after $mR_0(f)$ 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 $T/m \geq R_0(f)$ by Theorem 11. Thus $T \geq mR_0(f)$. ◀

For our applications, however, we will need a strengthened version of this theorem, which we call a threshold direct sum theorem for R_0.

▶ **Theorem 14** (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 $T \geq kR_0(f)$.*

Proof. Let A be such an algorithm. We convert A into an algorithm B for solving f on inputs from μ. The algorithm B is very similar to the algorithm in the proof of Theorem 13: on input $x \sim \mu$, it generates $m - 1$ additional inputs from μ, shuffles them, and feeds them into A. The algorithm A uses an expected T queries, but since x is shuffled with the fake inputs, it gets queried only T/m times in expectation. Moreover, the algorithm A solves an expected k of the m inputs, so the expected number of times it solves x is k/m. This means B solves x with probability k/m.

Moreover, when B solves x, it also finds a certificate. So by Lemma 3, we get a zero-error algorithm with expected query complexity $(T/m)/(k/m) = T/k$. We conclude that $T/k \geq R_0(f)$, so $T \geq kR_0(f)$, as desired. ◀

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 15.** *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 algorithm for $(f \circ g)_{\text{sab}}$, and let T be the expected query complexity of A (maximized over all inputs). We turn A into an 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 11. 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's 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, by Theorem 14, 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 $T/\mathrm{RS}(g)$. It follows that the expected number of queries B makes to x is at most $T/\mathrm{RS}(g)$. Thus we have $\mathrm{RS}(f) \leq T/\mathrm{RS}(g)$, which gives $T \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$. ◀

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

▶ **Corollary 16.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a partial function. Then $R(f^n) \geq \mathrm{RS}(f)^n/3$.*

This follows straightforwardly from observing that $R(f^n) = R_{1/3}(f^n) \geq (1-2/3)\,\mathrm{RS}(f^n)$ (using Theorem 8) and $\mathrm{RS}(f^n) \geq \mathrm{RS}(f)^n$ (using Theorem 15).

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

▶ **Theorem 17.** *Let f and g be partial functions. Then $\overline{R}_\epsilon(f \circ g) \geq \overline{R}_\epsilon(f)\,\mathrm{RS}(g)$.*

Proof. The proof follows a similar argument to the proof of Theorem 15. 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 † entries, so all the sabotaged inputs that B generates turn into 0- or 1-inputs if A tries to query a $*$ or † in them.

Since A uses T queries, by Theorem 14, 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 $\overline{R}_\epsilon(f) \leq T/\mathrm{RS}(g)$, and thus $T \geq \overline{R}_\epsilon(f)\,\mathrm{RS}(g)$. ◀

Setting ϵ to 0 yields the following corollary.

▶ **Corollary 18.** *Let f and g be partial functions. Then $R_0(f \circ g) \geq R_0(f)\,\mathrm{RS}(g)$.*

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

▶ **Corollary 19.** *Let f and g be partial functions. Then $R(f \circ g) \geq R(f)\,\mathrm{RS}(g)/10$.*

This follows from $R(f \circ g) \geq \overline{R}_{1/3}(f \circ g) \geq \overline{R}_{1/3}(f)\,\mathrm{RS}(g) \geq R(f)\,\mathrm{RS}(g)/10$, where we used $\overline{R}_{1/3}(f) \geq R(f)/10$, which can be shown by error reduction and Markov's inequality.

Finally, we can also show an upper bound composition result for randomized sabotage complexity. We defer the proof to the full version of this paper.

▶ **Theorem 20.** *Let f and g be partial functions. Then $\mathrm{RS}(f \circ g) \leq \mathrm{RS}(f) R_0(g)$. We also have $\mathrm{RS}(f \circ g) = O(\mathrm{RS}(f) R(g) \log \mathrm{RS}(f))$.*

5 Composition with the index function

To prove the composition result, we require the strong direct product theorem for randomized query complexity that was established by Drucker [4].

▶ **Theorem 21** (Drucker)**.** *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 $\gamma^3 k R(f)/11$ queries has success probability at most $(1/2+\gamma)^k$, for any $\gamma \in (0, 1/4)$.*

The first step to proving the main result that $R(f \circ \textsc{Ind} \circ g) = \Omega(R(f) R(\textsc{Ind}) R(g))$ is to show that $R(\textsc{Ind} \circ g)$ equals $\mathrm{RS}(\textsc{Ind} \circ g)$ up to constants if the index gadget is large enough.

▶ **Theorem 22.** *Let f be a partial Boolean function, and let $m = \Omega(R(f)^{1.1})$. Then $\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(R(f) \log m) = \Omega(R(\textsc{Ind}_m) R(f))$.*

Moreover, if $f_{\mathrm{ind}}^{\oplus c}$ is the defined as the index function on $c + 2^c$ bits composed with f in only the first c bits, we have $\mathrm{RS}_1(f_{\mathrm{ind}}^{\oplus c}) = \Omega(cR(f))$ when $c = 1.1 \log R(f) + \Omega(1)$.

Proof. Consider what the inputs to $(\textsc{Ind}_m \circ f)_{\text{sab}}$ look like. 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 star). 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, the only stars in the input will be in one array cell, whose index is the output of the first $\log m$ copies of f.

We now convert an $\mathrm{RS}(\textsc{Ind}_m \circ f)$ algorithm into a randomized algorithm for $f^{\log m}$. First, using Markov's inequality, 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 $f^{\log m}$ evaluated on the first $n \log m$ 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 $f^{\log m}$ with probability at least $\mathrm{RS}(\textsc{Ind}_m \circ f)^{-1}/4$.

Next, Theorem 21 implies that for any $\gamma \in (0, 1/4)$, either A's success probability is smaller than $(1/2+\gamma)^{\log m}$, or else A uses at least $\gamma^3 (\log m) R(f)/11$ queries. This means either $\mathrm{RS}(\textsc{Ind}_m \circ f)^{-1}/4 \le (1/2+\gamma)^{\log m}$ or $2\,\mathrm{RS}(\textsc{Ind}_m \circ f) \ge \gamma^3 (\log m) R(f)/11$, which means

$$\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega\left(\gamma^3 \min\left\{\left(\frac{2}{1+2\gamma}\right)^{\log m}, R(f)\log m\right\}\right). \tag{2}$$

Now, we have

$$\left(\frac{2}{1+2\gamma}\right)^{\log m} = m^{\log(2/(1+2\gamma))} = m^{1-\log(1+2\gamma)} \ge m^{1-2(\log e)\gamma} \ge m^{1-3\gamma}. \tag{3}$$

If $m \ge (R(f)\log R(f))^{(1-3\gamma)^{-1}}$, the above is at least $R(f)\log R(f) = \Omega(R(f)\log m)$, which means $\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(\gamma^3 R(f)\log m)$.

Note that $(1-3\gamma)^{-1} \le 1+12\gamma$ for all $\gamma \le 1/4$. Setting $r = 13\gamma$, we get

$$m = \Omega(R(f)^{1+r}) \Rightarrow \mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(r^3 R(f)\log m) \tag{4}$$

for all r satisfying $r = O(1)$ and $r = \Omega(\log\log R(f)/\log R(f))$. Setting $r = 0.1$ gives the desired result. The lower bound on $\mathrm{RS}_1(f_{\text{ind}}^{\oplus})$ follows similarly once we observe that sabotaging the array cell indexed by the outputs to the c copies of f introduces only one asterisk or obelisk, so the above argument lower bounds RS_1 and not only RS. ◀

Finally, we can prove Theorem 1, more precisely stated as follows.

▶ **Theorem 23.** *Let f and g be (possibly partial) functions, and let $m = \Omega(R(g)^{1.1})$. Then $R(f \circ \textsc{Ind}_m \circ g) = \Omega(R(f)R(g)\log m) = \Omega(R(f)R(\textsc{Ind}_m)R(g))$.*

Proof. By Corollary 19, we have $R(f \circ \textsc{Ind}_m \circ g) \ge R(f)\,\mathrm{RS}(\textsc{Ind}_m \circ g)/10$. Combining with Theorem 22 gives $R(f \circ \textsc{Ind}_m \circ g) = \Omega(R(f)R(g)\log m)$, as desired. ◀

6 Lifting theorem

To establish the connection between lifting theorems, we start with the following lemma, which gives a sabotage lower bound in the communication complexity setting.

▶ **Lemma 24.** *Let f be a (possibly partial) Boolean function on n bits, and let G_b be the index gadget on $\{0,1\}^b \times \{0,1\}^{2^b}$, with $b = O(\log n)$. Then*

$$R^{\mathrm{cc}}(f \circ G_b) = \Omega\left(\frac{R_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b)}{\log n \log\log n}\right), \tag{5}$$

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

Proof. We'll 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'd 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 $f \circ G_b$. 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've 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 amplification. We can therefore find the right gadget with bounded error, using $O(R^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)$ bits of communication.

Once we've found the right gadget, we can certify its validity by having Alice send the right index to Bob, using b bits of communication. Since we found a certificate with constant probability, we can use Lemma 3 to turn this into a zero-error algorithm. Thus

$$R_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b) = O(b + R^{\mathrm{cc}}(f \circ G_b) \log n \log\log n). \tag{6}$$

Since $b = O(\log n)$, we get $R_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b) = O(R^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)$. ◀

Equipped with this lemma we can prove the connection between lifting theorems (Theorem 2), stated more precisely as follows.

▶ **Theorem 25.** *Suppose that for all partial Boolean functions f on n bits, we have*

$$R_0^{\mathrm{cc}}(f \circ G_b) = \tilde{\Omega}(R_0(f)) \tag{7}$$

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

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \tilde{\Omega}(R(f)). \tag{8}$$

The loss in the $\tilde{\Omega}$ for the R result is only $\log n \log\log^2 n$ worse than the loss in the R_0 hypothesis.

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

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})}{c \log c}\right). \tag{9}$$

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 $cR^{\mathrm{cc}}(f \circ G_{2b})$ bits of communication, times $O(\log c)$ since we must amplify the randomized protocol to an error of $O(1/c)$.

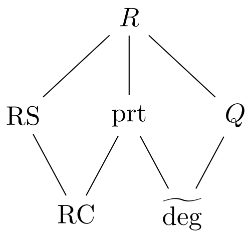

Figure 1 Lower bounds on $R(f)$.

Next, we apply Lemma 24 on $R^{\text{cc}}(f_{\text{ind}}^{\oplus c} \circ G_{2b})$ to get

$$R^{\text{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R^{\text{cc}}(f_{\text{ind}}^{\oplus c} \circ G_{2b})}{c \log c}\right) = \Omega\left(\frac{R_0^{\text{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}} \circ G'_{2b})}{c \log c \log n \log \log n}\right). \tag{10}$$

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 $(f_{\text{ind}}^{\oplus c})_{\text{usab}}$ 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 $(f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}}$ 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, so

$$R^{\text{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R_0^{\text{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}} \circ G'_{2b})}{c \log c \log n \log \log n}\right) = \Omega\left(\frac{R_0^{\text{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}} \circ G_b)}{c \log c \log n \log \log n}\right). \tag{11}$$

We're ready to use the assumed lifting theorem for R_0. To be more precise, let's suppose a lifting result that states $R_0^{\text{cc}}(f \circ G_b) = \Omega(bR_0(f)/\log^k n)$ for some integer k. Applying this to the above gives

$$R^{\text{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R_0^{\text{cc}}((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}} \circ G_b)}{c \log c \log n \log \log n}\right) = \Omega\left(\frac{bR_0((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}})}{c \log c \log^{k+1} n \log \log n}\right). \tag{12}$$

We note that

$$R_0((f_{\text{ind}}^{\oplus c})_{\text{usab}}^{\text{bin}}) = \Omega(R_0((f_{\text{ind}}^{\oplus c})_{\text{usab}})) = \Omega(\text{RS}_1(f_{\text{ind}}^{\oplus c})). \tag{13}$$

Setting $c = 1.1 \log R(f) + \Omega(1)$, we have $\text{RS}_1(f_{\text{ind}}^{\oplus c}) = \Omega(cR(f))$ by Theorem 22. Thus

$$R^{\text{cc}}(f \circ G_{2b}) = \Omega\left(\frac{bcR(f)}{c \log c \log^{k+1} n \log \log n}\right) = \Omega\left(\frac{bR(f)}{\log^{k+1} n \log \log^2 n}\right). \tag{14}$$

This gives the desired lifting theorem for R, with parameters at most $\log n \log \log^2 n$ worse than the assumed R_0 lifting theorem. ◀

7 Comparison with other lower bound methods

In this section we compare $\text{RS}(f)$ with other lower bound techniques for bounded-error randomized query complexity. Figure 1 shows the two most powerful lower bound techniques for $R(f)$, the partition bound ($\text{prt}(f)$) and quantum query complexity ($Q(f)$), 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 [2]. 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 show in this section. 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 $R_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$, showing that RS is a quadratically tight lower bound, even for zero-error randomized query complexity.

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 $R(f)$. Informally, what we show is that any separation between $R(f)$ and a lower bound measure like $Q(f)$, $\mathrm{prt}(f)$, or $\widetilde{\deg}(f)$ readily gives a similar separation between $\mathrm{RS}(f)$ and the same measure.

▶ **Theorem 26.** *There exist total functions f and g such that $\mathrm{RS}(f) \geq \mathrm{prt}(f)^{2-o(1)}$ and $\mathrm{RS}(g) = \widetilde{\Omega}(Q(g)^{2.5})$. There also exists a total function h with $\mathrm{RS}(h) \geq \widetilde{\deg}(h)^{4-o(1)}$.*

Proof. These separations were shown with $R(f)$ in place of $\mathrm{RS}(f)$ in [1] and [2]. To get a lower bound on RS, we can simply compose Ind with these functions and apply Theorem 22. 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 Theorem 22 implies that $\mathrm{RS}(f_{\mathrm{CS}}) = \widetilde{\Omega}(R(f))$ for all f (where f_{CS} denotes the cheat sheet version of f, as defined in [1]). In particular, cheat sheets can never be used to separate RS from R (by more than logarithmic factors).

7.2 Randomized certificate complexity

Randomized certificate complexity, $\mathrm{RC}(f)$, is a lower bound for $R(f)$ first studied in [?]. We can show that for any partial function f, randomized sabotage complexity upper bounds randomized certificate complexity.

▶ **Theorem 27.** *Let f be a partial function. Then $\mathrm{RS}(f) \geq \mathrm{RC}(f)/4$.*

We defer the definition of $\mathrm{RC}(f)$ and the proof of this theorem to the full version of the paper.

7.3 Zero-error randomized query complexity

▶ **Theorem 28.** *Let f be a total function. Then $R_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$ or alternately, $\mathrm{RS}(f) = \Omega(\sqrt{R_0(f)/\log R_0(f)})$.*

Proof. Let A be the $\mathrm{RS}(f)$ algorithm. The idea is 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 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 Markov's inequality. 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 block of x. Repeating this k times, we get a $2k\,\mathrm{RS}(f)$ query algorithm that queries a bit in any given block of x with probability at least $1-2^{-k}$.

Now, by [13], the number of 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 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 block is a certificate, by Lemma 3, 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 $R_0(f) = O(\mathrm{RS}(f)\,\mathrm{bs}(f)\log \mathrm{RC}(f))$. Since $\mathrm{bs}(f) \leq \mathrm{RC}(f) \leq \mathrm{RS}(f)$ by Theorem 27, we have $R_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$, or $\mathrm{RS}(f) = \Omega(\sqrt{R_0(f)/\log R_0(f)})$. ◀

References

1 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. *To appear in Proceedings of STOC 2016. arXiv preprint* `arXiv:1511.01937`, 2015.

2 Andris Ambainis, Martins Kokainis, and Robin Kothari. Nearly optimal separations between communication (or query) complexity and partitions. *To appear in Proceedings of CCC 2016. arXiv preprints* `arXiv:1512.00661` *and* `arXiv:1512.01210`, 2015.

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

4 Andrew Drucker. Improved direct product theorems for randomized query complexity. *Computational Complexity*, 21(2):197–244, 2012. `doi:10.1007/s00037-012-0043-7`.

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

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

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

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

9 Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Information Processing Letters*, 110(20):893–897, 2010. `doi:10.1016/j.ipl.2010.07.020`.

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

11 Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3-4):191–204, 1995. `doi:10.1007/BF01206317`.

12 Shelby Kimmel. Quantum adversary (upper) bound. In *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 557–568, 2012. doi:10.1007/978-3-642-31594-7_47.

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

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

15 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), July 2014. doi:10.4086/cjtcs.2014.006.

16 Denis Pankratov. Direct sum questions in classical communication complexity. Master's thesis, University of Chicago, 2012.

17 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

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

19 A. Yao. Probabilistic computations: Toward a unified measure of complexity. *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.