Space-Bounded Communication Complexity

Joshua Brody[1], Shiteng Chen[2], Periklis A. Papakonstantinou[2], Hao Song[2], and Xiaoming Sun[3]

[1]Computer Science Department, Aarhus University, Denmark
[2]Institute for Theoretical Computer Science, Institute for Interdisciplinary Information Sciences, Tsinghua University, China
[3]Institute of Computing Technology, Chinese Academy of Sciences, China

[joshua.e.brody | cst100 | ppapakons | baritono.tux | xiaoming.sun]@gmail.com

August 14, 2012

Abstract

In the past thirty years, Communication Complexity has emerged as a foundational tool to proving lower bounds in many areas of computer science. Its power comes from its generality, but this generality comes at a price—no superlinear communication lower bound is possible, since a player may communicate his entire input. However, what if the players are limited in their ability to recall parts of their interaction?

We introduce memory models for 2-party communication complexity. Our general model is as follows: two computationally unrestricted players, Alice and Bob, each have $s(n)$ bits of memory. When a player receives a bit of communication, he "compresses" his state. This compression may be an arbitrary function of his current memory contents, his input, and the bit of communication just received; the only restriction is that the compression must return at most $s(n)$ bits. We obtain memory hierarchy theorems (also comparing this general model with its restricted variants), and show super-linear lower bounds for some explicit (non-boolean) functions.

Our main conceptual and technical contribution concerns the following variant. The communication is one-way, from Alice to Bob, where Bob controls two types of memory: (i) a large, *oblivious* memory, where updates are only a function of the received bit and the current memory content, and (ii) a smaller, *non-oblivious/general* memory, where updates can be a function of the input given to Bob. We exhibit natural protocols where this semi-obliviousness shows up. For this model we also introduce new techniques through which certain limitations of space-bounded computation are revealed. One of the main motivations of this work is in understanding the difference in the use of space when computing the following functions: Equality (EQ), Inner Product (IP), and connectivity in a directed graph (Reach). When viewed as communication problems, EQ can be decided using 0 non-oblivious bits (and $\log_2 n$ oblivious bits), IP requires exactly 1 non-oblivious bit, whereas for Reach we obtain the same lower bound as for IP and conjecture that the actual bound is $\Omega(\log^2 n)$. In fact, proving that 1 non-oblivious bit is required becomes technically sophisticated, and the question even for 2 non-oblivious bits for any explicit boolean function remains open.

Keywords: communication complexity, space-bounded, memory-bounded

1 Introduction

In computer science research most lower bounds have an information theoretic nature, but for a few exceptions (e.g. time/space hierarchy theorems, and the recent ACC^0 lower bound [25]). These lower bounds amount to showing that there is an "information cost we must pay" if we wish to correctly complete the task. This is commonly proved either implicitly, or by a reduction an information theoretic setting such as the communication complexity. The Communication Complexity model, originally introduced by Yao [26], is almost information theoretic by definition. (In fact, establishing a strong formal connection between communication complexity and information theory is itself an active area of research; see e.g., [11, 4, 5] for more details.) In its simplest form two computationally unbounded players, Alice (who holds an input x) and Bob (who holds y), aim to evaluate a function $f(x, y)$ by communicating as little as possible. This is an area of intense research activity, has autonomous existence, and finds several applications in diverse areas such as circuit complexity ([16, 22]), VLSI design ([1, 27]), data structure lower bounds ([20, 21, 19]) streaming algorithms ([2, 15, 24, 12, 13]), and property testing ([7]) to name a just a few.

When lower bounds are derived by a reduction from communication complexity, the quality of the lower bound is bounded by the corresponding communication lower bound. In particular, in the classical communication complexity setting no super-linear communication lower bound is possible, as one player may simply send his entire input to the other.

To put things in context, consider the following example from streaming algorithms. Let $\textsc{Reach}(G, s, t)$ be the problem that, given a directed graph G, determines if t is reachable from s. For a graph of n vertices and a machine with working memory $O(\log n)$, classical communication complexity techniques show that the number of passes must be $\Omega(\frac{n}{\log n})$. However, if one believes the conjecture that Reach cannot be computed in logarithmic space, the actual lower bound on the number of passes should be infinite.

What if we modify the original communication complexity model in a way that Alice and Bob can use only a small amount, say $O(\log n)$, of memory between steps of communication? In this setting, when players receive a bit of communication, they may spend an arbitrary amount of computation deciding what to save in their memory; however, they still must *compress* the information they know at each step; i.e., they must take their $s(n)$ bits of memory and the communication just received and again save only $s(n)$ bits of memory. In this case super-linear communication lower bounds are possible, and furthermore common reduction arguments (including the one mentioned above) go through *intact*. Being able to prove such lower bounds sounds too good to be true. Simply restricting the memory of the players corresponds to a model at least as strong as width-bounded branching programs and depth-bounded circuits. Although slightly super-linear communication lower bounds is within the reach of known techniques, showing lower bounds that shed actual light on the limitations of space-bounded computation seems non-trivial. In this paper we obtain strong lower bounds for explicit non-boolean functions in the general model, and for explicit boolean functions in a certain restricted model. We discuss this restriction in Section 1.2 and view it as an important part of our conceptual contribution.

1.1 Background

Lam, Tiwari and Tompa [18] were the first to study tradeoffs between communication and space requirements when computing functions in the *straight-line protocols* model. Alternatively, a straight-line protocol can be modeled as a pebble game on an arithmetic or boolean circuit. On the arith-

metic model, they proved communication-space tradeoffs for matrix multiplication and polynomial convolution, through a detailed analysis of the structure of arithmetic circuits computing bilinear forms; i.e. the results rely heavily on the specifics of arithmetic circuits. On the other hand, the tradeoff for matrix-vector multiplication in their one-way boolean model follows a more generic information theoretic technical approach. Beame, Tompa and Yan [6] took a step further, introducing the more general *communicating branching programs* model. Adopting earlier techniques of Borodin et al. [9, 8] on branching programs into the communication setting, they showed stronger communication-space tradeoffs for matrix-vector multiplication and other functions.

In [18] and [6], techniques from classical computational complexity literature were modified to find application in the more powerful communication setting. Our motivation is in the opposite direction. We introduce a conceptually simple, purely information theoretic model. This choice of the model is not only a matter of elegance. First, given that currently we understand very little about computation itself it is not clear that there is any gain by bounding the players computationally. Second, this level of generality makes more transparent possible applications.

Besides [18, 6], some recent papers appear to be relevant to our approach. Klauck et al. [17] studied somewhat related communication-space tradeoffs in the randomized and quantum setting. Impagliazzo and Williams [14] studied a variant of communication complexity where players share synchronized access to a common clock. They showed that with the help of this synchronized clock, it is sometimes possible to save communication. This clock resembles a special type of memory we study in this paper called "oblivious memory". The *garden-hose* model, introduced recently by Buhrman et al. [10], also resembles several characteristics of the space-bounded communication complexity model we study here.

1.2 Oblivious Memory Updates – Compressing Interaction Obliviously

In addition to considering communication problems where players have limited space, we wish to understand an aspect of how space gets used in computing different functions of interest, e.g. the equality function EQ, inner product IP, and REACH.

Consider a one-way space-bounded protocol for computing equality in the two-party communication setting. Alice sends her input $x \in \{0,1\}^n$ bit-by-bit to Bob. Bob in turn compares each bit x_i to the corresponding bit y_i of his input $y \in \{0,1\}^n$. If Bob discovers a mismatch he outputs 0 and the protocol halts. Otherwise Bob halts after n bits of communication and outputs 1. We emphasize that in this protocol the players only need to maintain a counter which increases independent of their inputs.

A similar strategy also applies to the inner product function IP, where players receive n-bit strings and must compute their inner product modulo 2. Again, Alice and Bob keep counters incremented at each step, and Alice sends x to Bob bit-by-bit. This time, Bob keeps an additional bit to store the intermediate result. Namely, at step k Bob maintains a single bit containing the value $\sum_{i=1}^{k} x_i y_i$.

While both functions can be computed with $n + O(1)$ bits of communication and $\log_2 n + O(1)$ bits of work memory (optimal in the sense that both functions have deterministic communication complexity n), there is a difference in the way Bob uses his work memory. In the protocol for equality, the content of Bob's memory remains independent of his input, while in the protocol for inner product, the extra bit Bob uses depends on y. We will call the part of Bob's work memory whose content does not depend on y as "oblivious", and the part of Bob's work memory whose content does depend on y as "non-oblivious" or "general".

How about functions that have been conjectured to be hard for reasonably large space in the Turing Machine world? For example, we conjecture that $\textsc{Reach}(G, s, t)$ requires $\Omega(\log^2 n)$ *non*-oblivious bits, unless the oblivious memory is large enough to hold Alice's entire input. Interestingly, so far we were only able to prove that at least 1 non-oblivious bit is required, and in fact just proving such a bound becomes challenging.

Apart from the above natural semi-oblivious protocols there are more high-level reasons this two-type memory merits consideration. By allowing a large enough oblivious memory we can realize protocols where players are able to perform simple tasks such as counting and communicating small parts of the information given in their input. This allows us to consider a weaker "type" of memory and attempt to study the general one *in isolation*. Note that if we instead only had non-oblivious memory then on one hand memory size lower bounds of e.g. $\frac{\log_2 n}{3}$ are trivial (even counting is impossible in this amount of space), whereas lower bounds for slightly larger memory directly imply strong circuit lower bounds. Even proving lower bounds for a smaller number of non-oblivious bits turns out to be a daunting task. One reason is because oblivious bits are present, and oblivious bits aren't as weak as one might think. For instance, in Section 3, we show that you cannot replace oblivious memory by a slightly smaller amount of non-oblivious memory.

1.3 Our Techniques and a Roadmap to Our Results

In this work, we define the space-bounded communication complexity model and give upper bounds for many interesting problems. Importantly, we also provide lower bounds and create new proof techniques tailored to space-bounded communication. Our results are organized as follows. In Section 3 we give several robust memory hierarchy theorems. In particular, we show:

Theorem 1 (Memory Hierarchy Theorem, informally stated)**.** *For all $s(n)$, there are functions computable using a $(s(n) + log(n))$-space communication protocol that cannot be computed by any $s(n)$-space communication protocol.*

We also show a direct-sum type of upper bound, where oblivious protocols can be combined without losing the oblivious nature of the protocol. For example, this shows that 2-output-bit equality (where each player has two strings, and players must determine whether each pair is equal) can be computed with a logarithmic amount of oblivious memory. We present additional interesting examples of space-bounded protocols in Section 4.

In Section 5, we show a non-computability result for the general memory model for explicit functions with large outputs. Our first such function, All-EQ, computes equality on every subset of bits. The output of All-EQ is thus 2^n bits long. By using ideas from combinatorial designs, we create another function EQ-with-Design$_k$ whose output is polynomial in the number of input bits. We show that computing this function is also hard in the space-bounded communicaion model.

Theorem 2 (Lower bounds for non-boolean functions, informally stated)**.** *Computing* All-EQ *requires $\Omega(n)$ space in the space-bounded communication model. Computing* EQ-with-Design$_k$ *requires $\Omega(k \log n)$ space.*

Both lower bounds are close to their respective known upper bounds. A significant part of this work refers to new model-specific techniques we devised to prove lower bounds. The most involved such technique is in the proof that computing Inner Product requires non-oblivious memory.

Theorem 3. *Any one-way oblivious protocol for* IP *requires $\Omega(n)$ space.*

Proving this theorem using standard communication lower bound techniques is not possible, as these techniques do not account for the space used by players. To that end, we develop a way to combine the standard notion of a communication matrix with an accounting of the current state of memory at any point in time in a space-bounded protocol. We analyze how the output decision is made based on the changing state of memory through the course of a protocol and argue that with limited space, little progress can be made. To show this, we introduce a geometric/covering notion of progress. Making this precise, and quantifying the details is technical and involved. We develop these concepts in depth in Section 6 and provide a full proof in Section D.

2 The Space-Bounded Communication Complexity Model

For this paper, we focus on two-player deterministic protocols. Unless otherwise specified, we assume Alice and Bob receive inputs $x, y \in \{0,1\}^n$ and wish to compute a boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

In this section, we formally define the space-bounded communication complexity model as well as some variants. A player is space-bounded if he has a variable $M \in \{0,1\}^s$ corresponding to a limited amount of memory. A player's actions are defined by a *transition function*

$$T : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^s \times \{0,1\} \times \{0,1,\perp\} \ ,$$

where $T(b,x,m) = T(m',b',h)$ means the player, given input x, old memory contents m, and after receiving a bit of communication b, sets his memory contents to m', sends b' to the other player, and if $h \neq \perp$, the player halts and outputs h. A protocol is space-bounded when both players are space-bounded. Thus, a protocol can be described by a tuple (T_A, T_B).

Definition 4. *A* space-bounded communication protocol $\mathcal{P}$ *with s bits of memory is a tuple (T_A, T_B) where $T_A, T_B : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^s \times \{0,1\} \times \{0,1,\perp\}$, and in this paper, we use $M_A, M_B \in \{0,1\}^s$ to denote the content of Alice's and Bob's local memory, respectively. We say that $\mathcal{P}$ computes f if for all x, y, both players halt and output $f(x,y)$. The* communication cost *of $\mathcal{P}$ is the maximum total number of bits communicated over all inputs.*

Remark. Note that no restriction is placed on the computational power required to compute T_A and T_B. In particular, players may use an arbitrary amount of time *or space* to compute T_A or T_B. Nevertheless, between steps of communication, memory remains bounded.

It is helpful to consider how players interact during the course of a protocol. Initialily, Alice and Bob receive inputs x and y and initialize their memory to zero. Communication proceeds in a number of steps, where Alice receives b_B from Bob and applies

$$(M_A, b_A, h_A) \leftarrow T_A(b_B, x, M_A) \ .$$

Alice then sends b_A to Bob, who computes

$$(M_B, b_B, h_B) \leftarrow T_B(b_A, y, M_B) \ .$$

completing a step of communication. (Initially, Alice sets b_B to zero.) Players proceed until each halts and outputs $f(x,y)$.

We are particularly interested in bounds on the minimum amount of space required to compute a function, and in which classes of functions are computable in limited space.

Definition 5. $\mathsf{SPACECC}(s)$ *is the set of boolean functions computed by a space-bounded communication protocol using* s *bits of space.*

2.1 One-Way Semi-Oblivious Protocols

As mentioned in the introduction, we believe it is interesting to consider memory that is oblivious when communication is one-way.

Definition 6. *In a* one-way *protocol, Alice feeds a stream of bits to Bob, but Bob cannot communicate back. Each of them has* s *bits of memory.* $\mathsf{SPACECC}_{\rightarrow}(s)$ *is the set of boolean functions computable with one-way protocols using* s *bits of space.*

In an oblivious protocol, Alice is space-bounded as before, but Bob's memory is further restricted.

Definition 7. *In a one-way* semi-oblivious *protocol, Bob has two variables* $M_B^f \in \{0,1\}^{s_f}$ *and* $M_B^o \in \{0,1\}^{s_o}$ *and transition functions*

$$T_B^f : \{0,1\} \times \{0,1\}^n \times \{0,1\}^{s_f} \times \{0,1\}^{s_o} \to \{0,1\}^{s_f} \times \{0,1,\perp\}$$

and

$$T_B^o : \{0,1\} \times \{0,1\}^{s_o} \to \{0,1\}^{s_o}$$

The total amount of space used by Bob is $s_f + s_o$*. The protocol is* oblivious *if* $s_f = 0$*.*

$\mathsf{SPACECC}^{\mathsf{o}}_{\rightarrow}(s)$ *is the set of all functions computable by oblivious protocols where each player uses* s *bits of space.*

Bob uses T^f and T^o in the natural way. Note that updates to a player's oblivious memory are independent of both his input and his nonoblivious memory. Also note that in an oblivious protocol, the function T_B^f serves the purpose of deciding Bob's final answer (and that purpose only).

2.2 Space-bounded communication as State Machines.

It is natural to view a space-bounded communication protocol as the interaction between communicating state machines. In this view, Alice and Bob construct state machines $\mathcal{M}_x$ and $\mathcal{M}_y$ during a preprocessing stage. Each machine has a state for each possible memory configuration, plus two additional halt states YES and NO. From each non-halt state, there are 0 and 1 edges corresponding to the bit received from the other player. Each edge is further labeled by the output bit sent to the other player. Note that players construct different state machines on different inputs, since their behavior during a protocol differs on different inputs.

3 Basic Properties

We give a list of some basic properties of our space-bounded communication model and its variants. Observe that, as in other common space-bounded computation models, if a protocol runs for long enough then it repeats a configuration and does not halt.

Proposition 8. *Let* $\mathcal{P}$ *be a communication protocol where the players have memory space* $s(n)$ *and the protocol halts. Then, on every input* $(x,y) \in \{0,1\}^n \times \{0,1\}^n$ $\mathcal{P}$ *runs in at most* $2^{2s(n)+1}$*.*

3.1 Memory Hierarchy Theorem

A natural, initial question for these memory models is whether more space can be used to compute more functions.

Theorem 9. *For any $s(n) < \frac{n}{5}$, almost every boolean function (on two n-bit inputs) that can be computed with one-way oblivious protocol of work memory size $s(n) + \log n$, is not computable by any protocol (not necessarily oblivious) of work memory size $s(n)$.*

To prove this theorem, we will first prove Lemma 10 and Lemma 11 below.

Lemma 10. *The number of different boolean functions (on inputs from $\{0,1\}^n \times \{0,1\}^n$) that can be computed with protocols with memory size $s(n)$ is at most $2^{(4+s(n))\cdot 2^{n+s(n)+2}}$.*

Lemma 11. *The number of different boolean functions (on inputs from $\{0,1\}^n \times \{0,1\}^n$) that can be computed with one-way oblivious protocols of work memory size $s(n)$ ($s(n) \leq n$) is at least $2^{2^{n+s(n)}}$.*

Proof of Theorem 9. This follows immediately by comparing the lower bound on the number of boolean functions computable by one-way oblivious protocols of memory size $s(n)+\log n$ (Lemma 11) and the upper bound on the number of boolean functions computable by two-way general protocols of memory size $s(n)$ (Lemma 10). □

Note that this implies two hierarchy theorems, one for the general (not necessarily one-way or oblivious) space-bounded communication model, and one for the one-way oblivious model.

Corollary 12. *For any $s(n) < \frac{n}{5}$, $\mathsf{SPACECC}(s(n)) \subsetneq \mathsf{SPACECC}(s(n) + \log n)$.*

Corollary 13. *For any $s(n) < \frac{n}{5}$, $\mathsf{SPACECC}^{\mathsf{o}}_{\rightarrow}(s(n)) \subsetneq \mathsf{SPACECC}^{\mathsf{o}}_{\rightarrow}(s(n) + \log n)$.*

3.2 Parallel repetitions of one-way Oblivious Protocols

Let us revisit the one-way oblivious protocol for equality (Section 1.2). First, observe that the protocol may halt at different steps for different input pairs (x, y). This can be shown to be essential (see Section E). Now, consider the question of computing the non-boolean "2-bit EQ". In this problem Alice is given two n-bit strings x_1, x_2, and Bob is given y_1, y_2, and we want to find out if x_1 is equal to y_1, and simultaneously, if x_2 is equal to y_2. Interestingly, we do not need an extra non-oblivious bit to compute this 2-bit EQ. In fact, there is a more general property according to which we can compose "parallel" oblivious protocols on independent inputs.

Proposition 14. *For a pair of boolean functions f_1 and f_2, if each is a function on two n-bit inputs, $f_1, f_2 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and can be computed by a one-way oblivious protocol with space-bound $s_1(n), s_2(n)$ and communication cost $c_1(n), c_2(n)$, then the composite function $f(x, y) = (f_1(x, y), f_2(x, y))$ can be computed with a one-way oblivious protocol with space-bound $s_1(n) + 3s_2(n) + 1$ and communication cost $c_1(n) \cdot (c_2(n) + 2)$.*

4 Example Protocols

In this section, we briefly present some space-bounded protocols for two natural problems on directed graphs. We defer proofs of the theorems in this section to Section B.

Depth First Search on a Tree. The input to this problem is a directed binary tree $T = (V, E)$ with distinguished vertices $s, t \in V$, and we wish to determine which vertex is reached first in a depth-first lexicographic traversal of T. We refer to this problem as DFS-Tree and define DFS-Tree$(T, s, t) = 1$ if and only if s precedes t. In the space-bounded communication version, Alice and Bob are given different sides of a fixed vertex cut $A \uplus B = V$. Alice receives all edges leaving vertices in A, and Bob receives all edges leaving vertices in B.

Theorem 15. DFS-Tree $\in \mathsf{SPACECC}(\log n + \log\log n + O(1))$.

Note that while the space in the protocol achieving Theorem 15 is extremely limited, it requires worst-case communication $\Omega(n^2 \log(n))$, close to the theoretical maximum of $\Theta(n^2 \log^2 n)$ given by Proposition 8 and Theorem 15. We are interested in knowing if this is required, or if in general it is possible to *compress* communication close to this maximal bound. More generally we ask if there is a general scheme to bring the worst case communication cost of a space-bounded communication protocol down to $2^{s+o(s)}$ (from 2^{2s}).

Reachability. The input for this problem is again a directed graph $G = (V, E)$, with distinguished vertices s, t. Define Reach$(G, s, t) = 1$ if there exists an $s \rightsquigarrow t$ path in G; otherwise, Reach$(G, s, t) = 0$.

In the space-bounded communication version of this problem, players are again given different sides of a fixed vertex cut $A \uplus B = V$. Alice gets as input all edges in A; Bob gets as input all edges in B, and both players see all crossing edges. Let C_A denote the set of vertices in A adjacent to some vertex in B. Define C_B analogously. $C_A \cup C_B$ thus defines the boundary of the cut. Finally, let $C := \{s, t\} \cup C_A \cup C_B$, and let $c = |C|$. The performance of all of our protocols highly depend on c.

Theorem 16. Reach(G, s, t) *can be computed (i) using* $O(\log^2 c)$ *space and* $2^{O(\log^2 c)}$ *communication, (ii) using* $O(c)$ *space and* $O(c^2)$ *communication, (iii) with a one-way protocol that uses* $O(c^2)$ *space and* $O(c^2)$ *communication, and (iv) with a one-way protocol that uses* $O(c)$ *space and* $O(c^3)$ *communication.*

5 Lower Bound for explicit non-boolean functions

We give lower bounds for two explicit non-boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$. Note that some care must be made when discussing space-bounded communication for such functions. When the number of output bits is large, space lower bounds become trivial if we require players to output the entire function at once. Instead, we modify the model so players can output a *subset* of the output bits at any step. As long as the entire function is eventually output, and the answers are consistent, we say the protocol computes f.

Definition 17. *A space-bounded communication protocol* $\mathcal{P}$ *computing* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *using* s *bits of space is a tuple* (T_A, O_A, T_B, O_B)*, with*

- *transition functions* $T_A, T_B : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^s \times \{\text{HALT}, \perp\}$,
- *output functions* $O_A, O_B : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1,*\}^m$.

We use $M_A, M_B \in \{0,1\}^s$ to denote Alice and Bob's memory, respectively. $\mathcal{P}$ computes f if (i) both players halt on all inputs, (ii) for all $j \in [m]$, some player correctly outputs $f_j(x,y)$, and (iii) no player incorrectly outputs $f_j(x,y)$.

In this definition, if, e.g., the jth bit of $O_A(b,x,m)$ is $b \neq *$, then Alice outputs $f_j(x,y) = b$. Note that we allow players to (correctly) output bits of f multiple times. This allows players to compute f without having to remember which bits have already been computed.

Our first non-boolean function ALL-EQ : $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2^n}$ computes equality on all *subsets* of bits. Let $\{I_1, \ldots, I_{2^n}\}$ enumerate all possible subsets of $\{1, \ldots, n\}$. Then, the ith output bit of ALL-EQ is defined as:

$$\text{ALL-EQ}_i(x,y) := \begin{cases} 1 & \text{if } \forall j \in I_i,\ x_j = y_j \\ 0 & \text{otherwise} \end{cases}$$

The function EQ-WITH-DESIGN$_k$ is similarly defined, except that instead of using all 2^n the subsets of $\{1, 2, \ldots, n\}$, we use a combinatorial design containing only p^k many such subsets, where p is a prime number satisfying $p^2 = n$,[1] and k is any positive integer. This family of subsets satisfies the following two properties: each subset has size p, and the intersection of any two subsets has size at most k.[2]

Theorem 18. *For any $s(n) < \log_2(1.5) \cdot n - \log_2 6$, ALL-EQ $\notin$ $\mathsf{SPACECC}(s(n))$. ALL-EQ $\in$ $\mathsf{SPACECC}(n + O(1))$.*

Theorem 19. *For any constant positive integer k, positive number ϵ, and any $s(n) < (\frac{1}{2} - \epsilon)k\log(n)$, we have* EQ-WITH-DESIGN$_k$ $\notin$ $\mathsf{SPACECC}(s(n))$ *and* EQ-WITH-DESIGN$_k$ $\in$ $\mathsf{SPACECC}\left(\frac{1}{2}(k+1)\log(n) + O(1)\right)$.

In this section, we sketch the proof of Theorem 18, which is technically simpler. The proof of Theorem 19 uses similar techniques and appears in Section C.

As explained in Section 2.2, given a space-bounded communication protocol $\mathcal{P}$, for every possible input x to Alice, she has a corresponding state machine $\mathcal{M}_x$, and for every possible input y to Bob, he has a corresponding state machine $\mathcal{M}_y$. Connecting $\mathcal{M}_x$ and $\mathcal{M}_y$, let $\pi(\mathcal{M}_x, \mathcal{M}_y)$ denote the resulting computational history, and $\pi(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}$ the outputs of Alice and Bob. Let $\|\pi_A(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1$ and $\|\pi_B(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1$ denote the number of 1 output bits produced by Alice and Bob respectively.

Lemma 20. *If there is a protocol $\mathcal{P}$ that correctly computes* ALL-EQ*, then among the state machines $\{\mathcal{M}_x\}_{x\in\{0,1\}^n}$ and $\{\mathcal{M}_y\}_{y\in\{0,1\}^n}$, one of the following things must be true*

- *$\exists x \in \{0,1\}^n$, such that $\sum_{y\in\{0,1\}^n} \|\pi_A(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq \frac{3^n}{2}$*
- *$\exists y \in \{0,1\}^n$, such that $\sum_{x\in\{0,1\}^n} \|\pi_B(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq \frac{3^n}{2}$*

[1] We assume for simplicity that n is always the square of some prime number. Strictly speaking, it suffices to choose a prime number between $\lceil\sqrt{n}\rceil$ and $2\lceil\sqrt{n}\rceil$. Bertrand's postulate guarantees this is possible.

[2] A specific construction utilizes the one-to-one correspondence between $\mathbb{F}_p \times \mathbb{F}_p$ ($\mathbb{F}_p$ being the prime field of size p) and $\{1, 2, \ldots, n\}$. Consider all polynomials of degree at most $k-1$ on $\mathbb{F}_p$, they are all of the form $q(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1 x + a_0$. There are p^k such polynomials, each of them can determine a subset of $\mathbb{F}_p \times \mathbb{F}_p$ of size p, namely $\{(0, q(0)), (1, q(1)), \ldots, (p-1, q(p-1))\}$. These corresponds to a family $\{I_i\}_{i=1,2,\ldots,p^k}$ of subsets of $\{1, 2, \ldots, n\}$ which we can use to define EQ-WITH-DESIGN$_k$.

Proof of Theorem 18. We show the contrapositive. Suppose that there is a protocol $\mathcal{P}$ with memory size bound $s(n)$ that correctly computes ALL-EQ. Assume without loss of generality that the clause of x in Lemma 20 is true, and the input value that makes this condition true is $x_0 = 0^n$.

Consider the states in $\mathcal{M}_{x_0}$, for every state $\gamma \in \{0,1\}^{s(n)}$ and every possible communication bit $b \in \{0,1\}$, denote the edge leading out of γ labelled b as $e(\gamma, b)$, denote the number of 1 output bits produced at $e(\gamma, b)$ as $o^1(\gamma, b)$, and denote the set of $y \in \{0,1\}^n$ such that $\pi(\mathcal{M}_{x_0}, \mathcal{M}_y)$ passes through edge $e(\gamma, b)$ as $Y(\gamma, b)$. For every $y \in Y(\gamma, b)$, $\|\text{ALL-EQ}(x_0, y)\|_1 \geq o^1(\gamma, b)$, thus $2^{D(x_0,y)} \geq o^1(\gamma, b)$, $D(x_0, y) \geq \log_2 o^1(\gamma, b)$. That means $\|Y(\gamma, b)\|_1 \leq 2^n / o^1(\gamma, b)$.

On the other hand,

$$\begin{aligned}
\sum_{y \in \{0,1\}^n} \|\pi_A(\mathcal{M}_{x_0}, \mathcal{M}_y)\|_1 &= \sum_{\gamma \in \{0,1\}^{s(n)}} \sum_{b \in \{0,1\}} \sum_{y \in Y(\gamma,b)} o^1(\gamma, b) \\
&= \sum_{\gamma \in \{0,1\}^{s(n)}} \sum_{b \in \{0,1\}} \|Y(\gamma, b)\|_1 \cdot o^1(\gamma, b) \\
&\leq 2^{s(n)} \cdot 3 \cdot 2^n
\end{aligned}$$

Therefore $3^n/2 \leq 2^{s(n)} \cdot 3 \cdot 2^n$, which implies $s(n) \geq \log_2(1.5) \cdot n - \log_2 6$. □

6 Equality, Inner Product, Reachability, and beyond...

When not viewed as communication problems, the computational complexity for equality (EQ) is EQ $\in \mathsf{AC}^0$, for inner product IP $\in \mathsf{NC}^1$ but not in AC^0, and for REACH we know that it can be done non-deterministically in $O(\log n)$ space (see e.g. [3] for definitions and conjectures). A famous conjecture states that REACH $\notin \mathsf{NC}^1$, in fact REACH $\notin \mathsf{LogSPACE}$. We wish to understand (at least partly) the difference in the use of space when computing these three functions, and we ask this question in our semi-oblivious model. We have already seen the following protocols.

Theorem 21. EQ $\in \mathsf{SPACECC}^{\circ}_{\rightarrow}(\log_2 n)$, IP *is computable with* 1 *non-oblivious bit and* $\lceil \log_2 n \rceil$ *oblivious bits, and* REACH $\in \mathsf{SPACECC}(O(\log^2 n))$.

The most involved technical contribution of our work is the following theorem, which states that the IP protocol is space-optimal; i.e. we need this one non-oblivious bit.

Theorem 22. IP $\notin \mathsf{SPACECC}^{\circ}_{\rightarrow}(n/8)$.

We observe that IP can be reduced to REACH through local preprocessing, implying that REACH also requires 1 non-oblivious bit. We conjecture that REACH requires $\Omega(\log^2 n)$ non-oblivious bits. Even the more modest goal of showing that REACH requires 2 non-oblivious bits is open.

The proof of Theorem 22 is given in Section D. Below we attempt to flesh-out a more general, model-specific technique, which also serves as a high-level description of this argument.

Overview of the IP lower bound Let us first recall the concept of *communication matrix* from classical communication complexity. We organize the correct answers of the function being computed on all possible input pairs (x, y) in a matrix where rows are associated with x and columns with y, and the (x, y) position contains the value of $f(x, y)$. A *monochromatic rectangle* is a sub-matrix where f has the same value for each entry.

The only property of IP that our proof technique uses is the fact that all the monochromatic rectangles in the communication matrix of IP have a relatively small size bound (for two n-bit inputs, ths bound is 2^{n+1}). In fact, the same proof works for every function enjoying this property.

The Progress Measure. We use the number of columns that are "partially solved" after a certain number of steps as a progress measure. A column is called "partially solved" after t steps if after t steps the protocol has output the correct value and halted on at least one position in this column. Since the number of steps in a halting protocol is bounded, if we obtain an upper bound on the amount of progress in a single step then we can conclude that at the end of the protocol run, the number of "partially solved" columns can not reach 2^n (the number of columns in the matrix). Handling this progress measure appropriately is a subtle technical issue.

Bands and the protocol matrix. Note that in a one-way oblivious protocol we can think of Alice as uploading Bob's memory.[3] If we fix the value Alice uploads to Bob's memory in a particular step (plus the one bit of communication of that step), we are actually fixing a subset of possible value of x, and therefore a subset of rows in the communication matrix. We call such a subset of rows as a "band". That means to capture the current state of Bob's memory, we only need to look at those "bands". In each step, Bob makes his output decision based solely on his current memory state, communication bit (which collectively correspond to a "band" in the communication matrix) and his input y (which corresponds to a column in the matrix).

The Stepwise Upper Bound on Progress. If a "band" is too "narrow" (containing too few rows), we can somehow ignore it as being insignificant. On the other hand, if a "band" is too "wide" (containing too many rows), its contribution to the progress measure we discussed above is limited, due to the monochromatic rectangle size bound. That limit implies the desired upper bound on progress to complete the proof. Note that without excluding the narrow bands (and showing that there can't be too many) the argument breaks down.

What's left to be done? Conceptually, we introduce memory models for communication complexity, and technically we give model-specific, non-trivial arguments giving leverage to these new models. Many questions raised in this work are left open. Closing the $\log n$ gap between the levels of the memory hierarchies is one such question.

An important open question is towards devising a technique for showing explicit space lower bounds for 2 or more non-oblivious bits. How far can we push such a lower bound for Reach?

One issue we haven't touched at the current stage of development is what happens in the presence of randomness. Studying such variants opens the possibility for answering some restricted forms of open questions in Communication Complexity itself; e.g. proving strong direct-sum theorems but in a space-bounded setting.

Finally, there are all sorts of questions relating these new models to open problems in Computational Complexity. How does the semi-oblivious model relate to circuit complexity classes such as AC and $\mathsf{P/poly}$? We know that oblivious refinements of the Karchmer-Wigderson games do not provide something meaningful (a discussion about this will appear in the full version), but other possibilities are open. In general we would like to know if there are any other, genuine applications of the semi-oblivious model in other areas? One can easily devise not-so-natural settings where oblivious memory is relevant, but how about the natural settings people actually care about?

[3] See Lemma 23 in Section D.

A Proofs of the Basic Properties

Proof of Proposition 8. With the two players in protocol $\mathcal{P}$ combined (both M_A and M_B), they have $2s(n)$ bits of memory, plus the bit received by Alice at the beginning of a step, the system would have at most $2^{2s(n)+1}$ different configurations in total. Therefore, if $\mathcal{P}$ makes more than $2^{2s(n)+1}$ a configuration is repeated and $\mathcal{P}$ never halts. □

Proof of Lemma 10. According to Definition 4, a space-bounded protocol $\mathcal{P}$ is defined by functions T_A and T_B. For input size n and space-bound $s(n)$, the number of such function tuples is at most $2^{(4+s(n))\cdot 2^{n+s(n)+2}}$, each corresponds to one protocol. Each protocol can correctly compute at most one boolean function if any. Hence the conclusion. □

Proof of Lemma 11. Any boolean function that only depends on the first $s(n)$ bits of Alice's input x is computable by such protocols. [4] The number of such boolean functions is $2^{2^{n+s(n)}}$. This gives a lower bound on the number of boolean functions computable by such class of protocols. □

Proof of Proposition 14. Suppose the protocols for solving f_1 and f_2 are $\mathcal{P}_1 = (T_{A,1}, T^o_{B,1}, T^f_{B,1})$ (with Alice and Bob's memory denoted as $M_{A,1}$ and $M_{B,1}$ respectively), and $\mathcal{P}_2 = (T_{A,2}, T^o_{B,2}, T^f_{B,2})$ (with Alice and Bob's memory denoted as $M_{A,2}$ and $M_{B,2}$ respectively), respectively. Now we construct a protocol $\mathcal{P}$ for solving f. In this protocol, Alice has a $s_1(n)$-bit memory that corresponds to $M_{A,1}$, a $s_2(n)$-bit memory that corresponds to $M_{A,2}$, and an additional counter C_A of $2s_2(n)+1$ bits long. Similarly for Bob, we have $s_1(n)$-bit $M_{B,1}$, $s_2(n)$-bit $M^o_{B,2}$, $2s_2(n)+1$-bit C_B. And the protocol $\mathcal{P}$ works as follows (shown in Algorith 1 and 2):

Algorithm 1: Algorithm for solving $f(x,y) = (f_1(x,y), f_2(x,y))$, Alice's part

```
M_{A,1} ⟵ 0^n;
while YES do
    Alice simulate one step of P_1 by carrying out T_{A,1}, sending the computed bit to Bob;
    M_{A,2} ⟵ 0^n;
    C_A ⟵ 0;
    while C_A ≤ c_2(n) + 1 do
        Alice simulate one step of P_2 by carrying out T_{A,2}, sending the computed bit to Bob;
        increment C_A;
    end
end
```

It's easy to verify that if both $\mathcal{P}_1$ and $\mathcal{P}_2$ are one-way oblivious protocols with the specified parameters, $\mathcal{P}$ presented above is also a one-way oblivious protocol with the required parameters. Note that according to Proposition 8, if C_A and C_B are both $2s_2(n)+1$ bits long, they can count to $c_2(n)+1$. □

[4] The protocol works as follows: Alice and Bob both increment the content of their respective memory simultaneously step by step. Alice sends 0 to Bob until the content of Alice's memory coincides the first $s(n)$ bits of x, at which time Alice sends 1. Upon receiving a 1, Bob computes the final answer.

Algorithm 2: Algorithm for solving $f(x,y) = (f_1(x,y), f_2(x,y))$, Bob's part

$M_{B,1} \longleftarrow 0^n$;
while YES **do**
  Bob simulate one step of $\mathcal{P}_1$ by receiving one bit from Alice and carrying out $T^o_{B,1}$;
  $M_{B,2} \longleftarrow 0^n$;
  $C_B \longleftarrow 0$;
  while $C_B \leq c_2(n) + 1$ **do**
    Bob simulate one step of $\mathcal{P}_2$ by receiving one bit from Alice and carrying out $T^o_{B,2}$;
    if *both* $T^f_{B,1}$ *and* $T^f_{B,2}$ *give definitive output answers (not* $\perp$*)* **then**
      Bob gives the answer and halts;
    end
    increment C_B;
  end
end

B Example Protocols

Proof of Theorem 15. First, we consider an algorithm for solving this problem within $2\log(n) + O(1)$ bits of memory (see Algorithm 3). We emphasize that this algorithm is not the standard DFS algorithm. Later, we convert this algorithm into a space-bounded communication protocol.

In this algorithm, we only need to remember at any point in time the value of s', t' and s_d. In the space-bounded communication model, we construct a protocol where each player has $\log(n) + \log(\log(n)) + O(1)$ space, divided into three parts.

- Each player uses $\lceil \log(n) \rceil$ bits of memory to store s' *or* t'. When ever Alice maintains s', Bob keeps t', and vice versa. Whenever say, Alice needs to perform some operation on s' but currently holds t', she will inform Bob that she wants to exchange for s' with t', and they will exchange with the help of the second chunk of memory.
- Alice and Bob use the second chunk of $\lceil \log(\lceil \log(n) \rceil) \rceil$ bits of memory for an index into the first $\lceil \log(n) \rceil$ bits of memory. When they need to exchange or compare s' and t', they do so bitwise, using the index to remember which bit they are exchanging or comparing.
- An additional $O(1)$ bits per player is for housekeeping; e.g., to remember which step of the algorithm they are executing, who hold s' and who holds t', etc.

□

In the worst case scenario, the tree is highly imbalanced. Both s' and t' traverse $\Omega(n)$ vertices before they meet near the root of the tree, and furthermore, their traversal bounces back and forth between A and B, and so Alice and Bob end up exchange s' and t' in every step. In this case, the communication complexity of the protocol is $\Theta(n^2 \log(n))$. Given the space bound $s(n) = \log(n) + \log(\log(n)) + O(1)$, this communication complexity is close to the theoretical upper bound given by Proposition 8, which is $2^{2s(n)+2} = \Theta(n^2 \log^2 n)$. We are very interested to know if this close to $2^{2s(n)+2}$ worst case communication cost is "compressible", meaning the worst case communication cost can be brought down.

Algorithm 3: Space Efficient Algorithm for Solving the Decision Version of DFS

```
if s = t then output 0;
s' ⟵ s;
s_d ⟵ 0;
while s' ≠ r do
    if s' = t then output 1;
    t' ⟵ t;
    repeat
        t' ⟵ ParentOf(t');
        if t' = s' then output s_d;
    until t' = r;
    if s' = LeftChildOf(ParentOf(s')) then
        s_d ⟵ 0;
    else
        s_d ⟵ 1;
    end
    s' ⟵ ParentOf(s');
end
output s_d
```

Proof of Theorem 16. We give several different protocols for $\textsc{Reach}(G, s, t)$. Recall that in this problem, the inputs consists of a directed graph $G = (V, E)$ and distinguished vertices s, t. $\textsc{Reach}(G, s, t)$ outputs 1 if there exists an $s \rightsquigarrow t$ path in G.

In the communication version of this problem, Alice and Bob are again given different sides of a fixed vertex cut $A \uplus B = V$. Alice gets as input all edges in A; Bob gets as input all edges in B, and both players see all crossing edges. Let C_A denote the set of vertices in A adjacent to some vertex in B. Define C_B analogously. $C_A \cup C_B$ thus defines the boundary of the cut. Finally, let $C := \{s, t\} \cup C_A \cup C_B$, and let $c = |C|$. Given an instance G, s, t, let c denote the total number of vertices on the boundary of the cut. The performance of all of our protocols highly depend on c.

Our first protocol Savitch emulates Savitch's Theorem [23]. Note that if s and t are indeed connected in G, then trivially, there exists a path from s to t that uses at most c crossing edges. The key intuition in Savitch's Theorem is the following

> If u, v are connected using at most k crossing edges, then there exists w such that (i) u and w are connected using at most $k/2$ crossing edges and (ii) w and v are connected using at most $k/2$ crossing edges.

This intuition leads to a recursive algoirthm at the heart of Savitch's Theorem—determine if u, v are reachabile using at most k edges by recursively deciding if e.g. u, w are reachable using at most $k/2$ edges. In the protocol that emulates this theorem, Alice and Bob recursively implement this algorithm. This algorithm has $\log(c)$ depth. At each level of the algorithm, Alice and Bob use $O(\log c)$ space to iterate through w. At the bottom level of recursion, $O(c)$ communication occurs as players communicate whether e.g. w is reachable from u by crossing the cut zero times. The performance of Savitch is captured by the following theorem. Thus, Savitch costs $O(\log^2 c)$ space and uses $c^{O(\log c)}$ total communication.

In our second protocol BitVector, Alice and Bob iteratively maintain a vector $V \in \{0,1\}^c$, which represents which vertices on the boundary of the cut are reachable by s. For example, suppose $s \in A$. Alice then initializes V_u to be 1 for all u on her side of the cut boundary that are reachable from s. She sets $V_u := 0$ for all u on Bob's side of the cut boundary and for all u on her side that are not directly reachable from s. Bob then sets $V_w := 1$ for all w on his side of the cut boundary that are reachable from some u such that $V_u = 1$, again using only the vertices he sees. This proceeds iteratively until wither players learn that $\textsc{Reach}(G, s, t) = 1$, or until V doesn't change between messages. At this point, players output $\textsc{Reach}(G, s, t) = 0$ and halt the protocol. Overall, BitVector uses $O(c)$ space and $O(c^2)$ total communication.

This protocol is called Matrix and is similar to BitVector. This time, Alice constructs the "induced adjacency matrix" of C. Specifically, she sets $(u, v) = 1$ if v is reachable from u using only edges seen by Alice. Bob updates this matrix in a similar way—he sets $(u, v) = 1$ if there is a u, v path using only (i) the edges seen by Bob and (ii) paths seen by Alice. Note that this gives Bob enough information to compute $\textsc{Reach}(G, s, t)$ directly. In particular, Matrix uses a single round of communication. Matrix is a one-way protocol that uses $O(c^2)$ space and $O(c^2)$ total communication.

Our last reachability protocol is called MatrixMult. MatrixMult is a one-way protcol that is similar in spirit to Matrix, but uses less space, albeit the cost of a blowup in total communication.

Let M_A, M_B be the reachability matricies given Alice and Bob's inputs; e.g., $A_{uv} = 1$ if there is a $u \rightsquigarrow v$ path using only edges Alice sees. Let $M := M_A \vee M_B$ be the bitwise OR of these matrices. Alice and Bob solve Reach by computing the matrix product $M^* := M^{c+1}$ and outputting M^*_{st}. This value is 1 if there is an s, t path that crosses the cut only c times. The intuition behind the MatrixMult protocol is that M^*_{st} can be computed in a space-efficient manner. Specifically, Alice first sends the row of M_A containing s. Then, she sends all of A to Bob, column by column. This allows Bob to comput the row s of M^2. If Bob had enough space to store all of A, we would be done. Instead, Alice sends A to Bob $O(c)$ times, each time sending the elements column by column. In this way, Bob iteratively computes only row s of M^*. Thus, MatrixMult is a one-way protocol that uses $O(c)$ space and $O(c^3)$ total communication. □

C Proofs for Lower Bound Result of All-EQ and EQ-with-Design

Proof of Lemma 20. First let's introduce some notation, for two n-bit strings x and y, we denote the hamming distance between x and y as $H(x, y)$, and $D(x, y) = n - H(x, y)$, is the number of positions at which the corresponding bits are the same. $\|x\|_1$ is the number of 1 bits in x.

It's easy to see that

$$\|\textsc{All-EQ}(x, y)\|_1 = 2^{D(x,y)}$$

By definition, if a protocol $\mathcal{P}$ correctly computes All-EQ, then for every possible pair of inputs (x, y), $\|\pi_A(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 + \|\pi_B(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 = \|\textsc{All-EQ}(x, y)\|_1$.

Since $\mathcal{P}$ correctly computes All-EQ, then we have

$$\sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (\|\pi_A(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 + \|\pi_B(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1) = \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} |\textsc{All-EQ}(x, y)|$$

On the other hand

$$\begin{aligned}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}\|\textsc{All-EQ}(x,y)\|_1 &= \sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}2^{D(x,y)} = \sum_{x\in\{0,1\}^n}\sum_{j=0}^{n}\left(\sum_{y\in\{y_0|D(x,y)=j\}}2^{D(x,y)}\right)\\ &= \sum_{x\in\{0,1\}^n}\sum_{j=0}^{n}\binom{n}{j}2^j = \sum_{x\in\{0,1\}^n}3^n = 2^n\cdot 3^n\end{aligned}$$

Thus

$$\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}\|\pi_A(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 + \sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\|\pi_B(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 = 2^n\cdot 3^n$$

Therefore

- either $\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}\|\pi_A(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq 2^{n-1}\cdot 3^n$
- or, $\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\|\pi_B(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq 2^{n-1}\cdot 3^n$

Then, by averaging, we have

- either $\exists x\in\{0,1\}^n$, such that $\sum_{y\in\{0,1\}^n}\|\pi_A(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq \frac{3^n}{2}$
- or, $\exists y\in\{0,1\}^n$, such that $\sum_{x\in\{0,1\}^n}\|\pi_B(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq \frac{3^n}{2}$

□

Proof of Theorem 19. For the upper bound part, we give the following straightforward protocol: Alice and Bob each has two counters, one $\lceil k\log p\rceil$ bits long, to enumerate through the p^k subsets in the function definition; another one, $\lceil\log p\rceil$ bits long, to enumerate through the bits in a particular subset. In each step, they look at the counters, and compare the corresponding bits in their inputs x and y.

For the lower bound part, we show the contrapositive. Suppose that there is a protocol $\mathcal{P}$ with memory size bound $s(n)$ that correctly computes EQ-WITH-DESIGN. Using the notation introduced in the previous proof, we have

$$\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(\|\pi_A(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1 + \|\pi_B(\mathcal{M}_x,\mathcal{M}_y)|_{\mathsf{out}}\|_1) = \sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}|\textsc{EQ-with-Design}(x,y)|$$

Suppose the family of subsets of $\{1,2,\ldots,n\}$ we use to define EQ-WITH-DESIGN is $\{I_i\}_{i=1,2,\ldots,p^k}$. We have

$$\begin{aligned}\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}|\textsc{EQ-with-Design}(x,y)| &= \sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}|\{i\mid EQ_{I_i}(x,y)=1\}|\\ &= \sum_{y\in\{0,1\}^n}\sum_{i\in\{1,2,\ldots,p^k\}}|\{x\mid EQ_{I_i}(x,y)=1\}|\\ &= \sum_{y\in\{0,1\}^n}\sum_{i\in\{1,2,\ldots,p^k\}}2^{p^2-p}\\ &= 2^n\cdot p^k\cdot 2^{p^2-p}\end{aligned}$$

By averaging like we did in the proof of Lemma 20, we have

- either $\exists x \in \{0,1\}^n$, such that $\sum_{y\in\{0,1\}^n} \|\pi_A(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq 2^{p^2-p-1} \cdot p^k$
- or, $\exists y \in \{0,1\}^n$, such that $\sum_{x\in\{0,1\}^n} \|\pi_B(\mathcal{M}_x, \mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq 2^{p^2-p-1} \cdot p^k$

Again, like we did in the proof of Theorem 18, we assume, without loss of generality, that the clause of x above is true, and the input value that makes this condition true is $x_0 = 0^n$.

$$\sum_{y\in\{0,1\}^n} \|\pi_A(\mathcal{M}_{x_0}, \mathcal{M}_y)|_{\mathsf{out}}\|_1 \geq 2^{p^2-p-1} \cdot p^k \tag{1}$$

And for every state $\gamma \in \{0,1\}^{s(n)}$ in the corresponding state machine $\mathcal{M}_{x_0}$, and for every possible communication bit $b \in \{0,1\}$, we likewise denote the edge leading out of γ labelled b as $e(\gamma, b)$, denote the number of 1 output bits produced at $e(\gamma, b)$ as $o^1(\gamma, b)$, and denote the set of $y \in \{0,1\}^n$ such that $\pi(\mathcal{M}_{x_0}, \mathcal{M}_y)$ passes through edge $e(\gamma, b)$ as $Y(\gamma, b)$.

$$\begin{aligned}
\sum_{y\in\{0,1\}^n} \|\pi_A(\mathcal{M}_{x_0}, \mathcal{M}_y)|_{\mathsf{out}}\|_1 &= \sum_{\gamma\in\{0,1\}^{s(n)}} \sum_{b\in\{0,1\}} \sum_{y\in Y(\gamma,b)} o^1(\gamma, b) &(2)\\
&= \sum_{\gamma\in\{0,1\}^{s(n)}} \sum_{b\in\{0,1\}} |Y(\gamma, b)| \cdot o^1(\gamma, b) &(3)
\end{aligned}$$

In addition, we define $\alpha(t) = \min_{\substack{i_1,i_2,\ldots,i_t\in\{1,2,\ldots,p^k\}\\ i_1,i_2,\ldots,i_t \text{ are all different}}} |\bigcup_{j=1}^t I_{i_j}|$, then

$$\forall\gamma \in \{0,1\}^{s(n)}, \forall b \in \{0,1\} \quad o^1(\gamma, b) \cdot |Y(\gamma, b)| \leq \max_{t\in\{1,2,\ldots,p^k\}} t \cdot 2^{p^2-\alpha(t)} \tag{4}$$

And clearly we have clearly we have

- $\alpha(t)$ is non-decreasing for $t \in \{1, 2, \ldots, p^k\}$. In particular, for $t \in \{1, 2, \ldots, p/2k\}$, $\alpha(t)$ is strictly increasing
- $\forall t \in \{1, 2, \ldots, p^k\} \quad \alpha(t) \geq tp - \binom{t}{2}k$

therefore for $t \in \{1, 2, \ldots, p/2k\}$, $t \cdot 2^{p^2-\alpha(t)}$ is strictly decreasing

$$\begin{aligned}
\max_{t\in\{1,2,\ldots,p^k\}} t \cdot 2^{p^2-\alpha(t)} &= \max\big(\max_{t\in\{1,2,\ldots,p/2k\}} t \cdot 2^{p^2-\alpha(t)}, \max_{t\in\{p/2k,p/2k+1,\ldots,p^k\}} t \cdot 2^{p^2-\alpha(t)}\big)\\
&\leq \max\big(t \cdot 2^{p^2-\alpha(t)}|_{t=1}, \left(\max_{t\in\{p/2k,p/2k+1,\ldots,p^k\}} t\right) \cdot \left(\max_{t\in\{p/2k,p/2k+1,\ldots,p^k\}} 2^{p^2-\alpha(t)}\right)\big)\\
&= \max(2^{p^2-p}, p^k \cdot \left(2^{p^2-\alpha(t)}\right)|_{t=p/2k})\\
&\leq \max(2^{p^2-p}, p^k \cdot 2^{p^2(1-7/16k)})\\
&= 2^{p^2-p}
\end{aligned}$$

Combine this with (1), (3), and (4), we have

$$\begin{aligned} 2^{s(n)+1} \cdot 2^{p^2-p} &\geq 2^{s(n)+1} \cdot \max_{t \in \{1,2,\ldots,p^k\}} t \cdot 2^{p^2-\alpha(t)} \\ &\geq \sum_{\gamma \in \{0,1\}^{s(n)}} \sum_{b \in \{0,1\}} |Y(\gamma, b)| \cdot o^1(\gamma, b) \\ &= \sum_{y \in \{0,1\}^n} \|\pi_A(\mathcal{M}_{x_0}, \mathcal{M}_y)|_{\mathsf{out}}\|_1 \\ &\geq 2^{p^2-p-1} \cdot p^k \end{aligned}$$

Therefore $2^{s(n)+1} \geq p^k/2$, $s(n) \geq (\frac{1}{2} - \epsilon)k \log(n)$ for any positive number ϵ and $p = \sqrt{n}$. □

D Proof of the Inner Product Non-Computability Result

Lemma 23. *Given a one-way semi-oblivious protocol $\mathcal{P}$ with Bob's oblivious work memory of size $s(n)$, there is a family of functions $\{K_i\}_{i \in \mathbb{N}^+} : \{0,1\}^n \to \{0,1\}^{s(n)} \times \{0,1\}$, such that for any positive integer $i \in \mathbb{N}^+$, and $x \in \{0,1\}^n$, $K_i(x)$ correctly computes the value of (M_B^o, b_A) at step i, in which M_B^o is the content of Bob's oblivious work memory if Alice and Bob executes $\mathcal{P}$ on input x and some $y \in \{0,1\}^n$, b_A is the bit Alice sends to Bob in such an execution.*

Proof. This follows immediately by Definition 7, where the value of M_B^o and b_A rely only on functions T_A and T_B^o. Both functions operate only on input x, independent of y. We conclude with an obvious induction. □

Proof of Theorem 22. We show that if a correct protocol for IP has no non-oblivious bits then the number of oblivious bits $s(n) \geq \frac{n}{8}$. Let $\mathcal{P}$ be a one-way oblivious protocol for IP of memory size $s(n)$. By definition only Bob using T_B^f is able to compute the correct answer to the given input.

By Lemma 23 there's a family of functions $\{K_i\}$ on $x \in \{0,1\}^n$ that would affect Bob's decision (in the sense that Bob's final output only depends on y and the output of K_i, if we denote the output of $K_i(x)$ as t, then T_B^f can be written as $T_B^f(y,t)$).

We construct a set $\mathcal{X} \subseteq \{0,1\}^n$ through the following process (Algorithm 4)

Algorithm 4: Construct $\mathcal{X} \subseteq \{0,1\}^n$

$\mathcal{X} \longleftarrow \{0,1\}^n$;
while $\exists t \in \{0,1\}^{s(n)+1}, i \in \{1,2,\ldots,2^{2s(n)+2}\}$ *such that* $|K_i^{-1}(t) \cap \mathcal{X}| < 2^{4s(n)}$ **do**
  $\mathcal{X} \longleftarrow \mathcal{X} \setminus K_i^{-1}(t)$;
end

It is not hard to see that this process terminates with an non-empty $\mathcal{X}$ if $s(n) < n/8$. Here is why. The following observations are obvious

- Any pair of (t,i) would be chosen at most once, $(t,i) \in \{0,1\}^{s(n)+1} \times \{1,2,\ldots,2^{2s(n)+2}\}$, thus there would be at most $2^{s(n)+1}(1 + 2^{2s(n)+2})$ iterations of the loop.
- During each iteration of the loop, the cardinality of $\mathcal{X}$ would decrease by at most $2^{4s(n)}$, since $\mathcal{X} \setminus K_i^{-1}(t) = \mathcal{X} \setminus (K_i^{-1}(t) \cap \mathcal{X})$ and $|K_i^{-1}(t) \cap \mathcal{X}| < 2^{4s(n)}$.

That means in the end the cardinality of set $\mathcal{X}$ is at least $2^n - 2^{4s(n)}2^{s(n)+1}(1+2^{2s(n)+2})$, which is positive given $s(n) < \frac{n}{8}$ and n is large enough.

Since $\mathcal{P}$ computes IP on $\{0,1\}^n \times \{0,1\}^n$ then it also computes IP on inputs from $\mathcal{X} \times \{0,1\}^n$. By inspection on the process which constructs $\mathcal{X}$, for every $t \in \{0,1\}^{s(n)+1}$ and $i \in \{1,2,\ldots,2^{2s(n)+2}\}$, we have either $|K_i^{-1}(t) \cap \mathcal{X}| \geq 2^{4s(n)}$ or $|K_i^{-1}(t) \cap \mathcal{X}| = 0$, define

$$\Gamma := \{(i,t) \mid |K_i^{-1}(t) \cap \mathcal{X}| \geq 2^{4s(n)}\}$$

Now consider the execution of the the protocol $\mathcal{P}$ on domain $\mathcal{X} \times \{0,1\}^n$. For $i \in \{1,2,\ldots,2^{2s(n)+2}\}$, define

$$Y_i := \{y \mid \exists x \in \mathcal{X} \text{ such that } \mathcal{P} \text{ have halted on } (x,y) \text{ before step } i\}$$

For $(i,t) \in \Gamma, j \in \{0,1\}$, define

$$L_{i,t}^{(j)} := \{y \mid y \notin Y_i \text{ and } T_B^f(y,t) = j\}$$

By definition we have

$$|Y_{i+1}| \leq |Y_i| + \sum_{t \in \{t' \mid (i,t') \in \Gamma\},\ j \in \{0,1\}} |L_{i,t}^{(j)}| \tag{5}$$

Since $\mathcal{P}$ gives correct answer for every pair $(x,y) \in \mathcal{X} \times \{0,1\}^n$, this means that for every $(i,t) \in \Gamma$ and every $y \in L_{i,t}^{(0)}$, we know $y \notin Y_i$, then

$$x \perp y, \quad \text{for every } x \in K_i^{-1}(t) \cap \mathcal{X}$$

In linear algebra terms, every vector in $L_{i,t}^{(0)}$ is orthogonal to a set of at least $2^{4s(n)}$ different non-zero vectors (in a n-dimensional space), thus $|L_{i,t}^{(0)}| \leq 2^{n-4s(n)}$.

Similarly for any $y \in L_{i,t}^{(1)}$, we have

$$\forall x \in K_i^{-1}(t) \cap \mathcal{X} \quad \mathrm{IP}(x,y) = 1$$

We know that $K_i^{-1}(t) \cap \mathcal{X} \neq \emptyset$, choose some $x_0 \in K_i^{-1}(t) \cap \mathcal{X}$, we have

$$\forall x \in (K_i^{-1}(t) \cap \mathcal{X}) \setminus \{x_0\} \quad x - x_0 \perp y$$

Thus $|L_{i,t}^{(1)}| \leq \frac{2^n}{2^{4s(n)}-1}$.

Substitute the upper bound we just obtained for $|L_{i,t}^{(0)}|$ and $|L_{i,t}^{(1)}|$ into equation (5) and considering the fact that $|Y_0| = 0$, we have

$$|Y_i| \leq i \cdot \frac{2^n}{2^{4s(n)}-1} \cdot 2 \cdot 2^{s(n)+1}$$

$$|Y_{2^{2s(n)}+1}| \leq (2^{2s(n)+2}+1) \cdot \frac{2^n}{2^{4s(n)}-1} \cdot 2 \cdot 2^{s(n)+1} < 2^n$$

That is, after $2^{2s(n)+2}+1$ steps, $Y_{2^{2s(n)+2}+1} \neq \{0,1\}^n$, there exist input pairs $(x,y) \in \mathcal{X} \times \{0,1\}^n$ on which the protocol $\mathcal{P}$ has not halted, and therefore (Proposition 8) it never halts. $\square$

E One-Way Oblivious Protocols for EQ

Proposition 24. *There is no one-way oblivious protocol that can correctly solve* EQ *within space size* $s(n) < n - 1$ *and halt at the same step for all input pairs* $(x, y) \in \{0,1\}^n \times \{0,1\}^n$.

Proof. We prove this by contradiction. Suppose there is such a protocol $\mathcal{P}$, which halts at some step i_0 for every input pair (x, y) and correctly computes EQ. As we have shown in Lemma 23 in Section D, Alice has this "upload function" K_{i_0} for step i_0. Since K_{i_0} takes a n-bit input, but gives a $(s(n)+1)$-bit output, $s(n)+1 < n$, there must be two different $x_1 \neq x_2$, such that $K_{i_0}(x_1) = K_{i_0}(x_2)$.

On the other hand, $\mathcal{P}$ correctly computes EQ, that means

$$\begin{aligned} T_B^f(x_1, K_{i_0}(x_1)) &= \mathrm{EQ}(x_1, x_1) = 1 \\ T_B^f(x_1, K_{i_0}(x_2)) &= \mathrm{EQ}(x_1, x_2) = 0 \end{aligned}$$

This contradicts with the fact that T_B^f is a deterministic function and $K_{i_0}(x_1) = K_{i_0}(x_2)$. □

References

[1] A. V. Aho, J. D. Ullman, and M. Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 133–139, Boston, MA, April 1983.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. Preliminary version in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 20–29, 1996.

[3] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, April 2009.

[4] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[5] B. Barak, M. Braverman, X. Chen, and A. Rao. How to compress interactive communication. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 67–76, 2010.

[6] P. Beame, M. Tompa, and P. Yan. Communication-space tradeoffs for unrestricted protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 420–428. IEEE Computer Society, October 1990.

[7] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, 2011.

[8] A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM Journal on Computing*, 11(2):287–297, 1982.

[9] A. Borodin, M. J. Fischer, D. G. Kirkpatrick, N. A. Lynch, and M. Tompa. A time-space tradeoff for sorting on non-oblivious machines. *Journal of Computer and System Sciences*, 22(3):351–364, June 1981.

[10] H. Buhrman, S. Fehr, C. Schaffner, and F. Speelman. The garden-hose game: A new model of computation, and application to position-based quantum cryptography. *CoRR*, 2011.

[11] A. Chakrabarti, Y. Shi, A. Wirth, and A. C. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 270–278, 2001.

[12] A. Gál and P. Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. In *Proceedings of the 48th Annual Symposium on Foundations of Computer Science*, pages 294–304, 2007.

[13] M. M. Halldórsson, X. Sun, M. Szegedy, and C. Wang. Streaming and communication complexity of clique approximation. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming*, pages 449–460, 2012.

[14] R. Impagliazzo and R. Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity*, pages 259–269, Cambridge, MA, June 2010. IEEE Computer Society.

[15] P. Indyk and D. Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science*, pages 283–289, 2003.

[16] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 539–550, Chicago, IL, May 1988.

[17] H. Klauck. Quantum and classical communication-space tradeoffs from rectangle bounds. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 384–395, Chennai, India, December 2004.

[18] T. W. Lam, P. Tiwari, and M. Tompa. Tradeoffs between communication and space. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 217–226, Seattle, Washington, May 1989.

[19] K. G. Larsen. The cell probe complexity of dynamic range counting. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 85–94, 2012.

[20] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 103–111, 1995.

[21] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 232–240, 2006.

[22] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, June 1992.

[23] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[24] X. Sun and D. P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 336–345, 2007.

[25] R. Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.

[26] A. C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 209–313, Atlanta, GA, April–May 1979.

[27] A. C. Yao. The entropic limitations of VLSI computations. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 308–311, Milwaukee, WI, May 1981.