Information Cost Tradeoffs for Augmented Index and Streaming Language Recognition

Amit Chakrabarti[1] Dartmouth College

Graham Cormode AT&T Labs–Research

Ranganath Kondapally[1] Dartmouth College

Andrew McGregor[2] University of Massachusetts Amherst

***Abstract*—This paper makes three main contributions to the theory of communication complexity and stream computation. First, we present new bounds on the information complexity of AUGMENTED-INDEX. In contrast to analogous results for INDEX by Jain, Radhakrishnan and Sen [*J. ACM*, 2009], we have to overcome the significant technical challenge that protocols for AUGMENTED-INDEX may violate the "rectangle property" due to the inherent input sharing. Second, we use these bounds to resolve an open problem of Magniez, Mathieu and Nayak [*STOC*, 2010] on the multi-pass complexity of recognizing Dyck languages. This results in a natural separation between the standard multi-pass model and the multi-pass model that permits reverse passes. Third, we present the first *passive memory checkers* that verify the interaction transcripts of priority queues, stacks, and double-ended queues. We obtain tight upper and lower bounds for these problems, thereby addressing an important sub-class of the memory checking framework of Blum et al. [*Algorithmica*, 1994].**

***Keywords*-augmented index, communication complexity, data streams, lower bounds, memory checking**

I. Introduction

In recent work, Magniez, Mathieu and Nayak [16] considered the streaming complexity of language recognition. That is, given a string σ of length n, what is the (randomized) space complexity of a recognizer for a language L that is allowed only sequential access to σ? This question can be viewed as a generalization of the classic notion of regularity of languages: one now considers automata that are allowed (1) randomization, and (2) a variable number of states that may depend on the input length. Their main result provided near-matching bounds for single-pass recognizers for DYCK(2), the language of properly nested parentheses of two kinds. In this paper, we look at the broader question and present the first *multi-pass* space lower bounds for several languages, including DYCK(2), resolving an open question of theirs. We also study the complexity of languages that arise in the context of *memory checking* [4], and present tight upper and lower bounds for them. Our key technical contributions rely on a new understanding of the information complexity of the augmented index problem, which leads to these multi-pass lower bounds.

[1]Work supported by NSF Grant IIS-0916565 and by Dartmouth College startup funds.

[2]Work supported by NSF CAREER Award CCF-0953754 and by University of Massachusetts startup funds.

Background, Augmented Index, and New Bound: The INDEX problem is one of a handful of fundamental problems in communication complexity [15]: Alice has a string $x \in \{0,1\}^n$, and Bob has an index $k \in [n]$; the players wish to determine the kth bit of x, written as x_k. It is easy to show that the problem is "hard"—requiring $\Omega(n)$ communication—when messages only go from Alice to Bob, and is "easy"—solvable using $O(\log n)$ communication—without this restriction. The lower bound holds for randomized protocols [1]. This makes INDEX the canonical hard-for-one-way, easy-for-two, communication problem. Is there really anything new to say on such a fundamental problem?

As it turns out, there is, provided one asks the right questions. Since INDEX is an asymmetric problem, it makes sense to ask for the best possible tradeoff between the number, a, of bits communicated by Alice, and the number, b, communicated by Bob. As shown by Miltersen et al. [17], we must have $a \geq n/2^{O(b)}$, and a simple two-round Bob→Alice→Bob protocol (with Bob announcing the output) shows that $a \leq \lceil n/2^b \rceil$ is achievable. A more nuanced question asks for the best tradeoff of *information revealed* by each player to the other in a protocol for INDEX, also called the *information costs* of Alice and Bob (we shall soon formally define these). In principle, this tradeoff could have been better, as it is possible for messages to reveal less information than their length. This issue was considered (in a more general quantum communication setting) by Jain, Radhakrishnan and Sen [13] who called this the "privacy tradeoff" for the problem, and showed that $a \geq n/2^{O(b)}$ still holds, where a and b now represent information costs.

Such an information cost tradeoff opens up interesting possibilities for applications to lower bounds for more complex problems, via the direct sum properties of this measure [3], [5]. One such application is the aforementioned DYCK(2) lower bound. However, the tradeoff theorem of Jain et al. is not strong enough to obtain the required direct sum result. One needs a tradeoff lower bound in a variant of INDEX where Alice and Bob have much more "help," in two ways. First, we relax INDEX so that Bob additionally gets to see the length-$(k-1)$ prefix of Alice's input; the resulting variant has been called AUGMENTED-INDEX [8], [10], [14] and the one-way communication lower bound easily extends to it [17]. Second, in our variant of AUGMENTED-INDEX, Bob also gets a *check bit* $c \in \{0,1\}$ and must verify that $x_k = c$. This second twist clearly does not matter when considering communication complexity, but for us it makes a huge difference, because our applications require that we measure information cost under an "easy" distribution, where x_k always equals c. With this background, we state our main theorem informally. A formal version appears as Theorem 4, after the necessary definitions.

Theorem 1 (Informal)**.** *In a randomized communication protocol for* AUGMENTED-INDEX *with constant two-sided error, either Alice reveals* $\Omega(n)$ *information about her input, or Bob reveals* $\Omega(1)$ *information about his, where information is measured according to an "easy" input distribution.*

The natural point of comparison is a similar theorem of Magniez et al. [16], that works only for restricted protocols: Alice must be deterministic (thus, her information cost is just her usual communication cost), and the protocol must be two-round with an Alice→Bob→Alice communication pattern. Under these conditions, for errors below $O(1/n^2)$, they show that either Alice sends $\Omega(n)$ bits, or else Bob reveals $\Omega(\log n)$ information. This theorem is not quite a special case of ours, because of the higher lower bound on Bob's information cost. This is inevitable: for a general communication pattern, one cannot obtain a tradeoff that strong, because of the aforementioned $a \leq \lceil n/2^b \rceil$ upper bound. However, we suspect that the optimum tradeoff lower bound is in fact of the form $a \geq n/2^{\widetilde{O}(b)}$, where the $\widetilde{O}$-notation hides factors polylogarithmic in b. We leave this conjectured generalization as an open problem.

Streaming Language Recognition: In the streaming model, we have one-way access to input and working memory sublinear in the input size N. Historically, problems considered have focused on estimating statistics. Recognizing structural properties of streaming strings is just as natural a problem and yet such language recognition problems have only recently been considered. It transpires that AUGMENTED-INDEX has a key role to play in proving bounds.

A first application is direct: following [16], a two-step argument shows $\Omega(\sqrt{N})$ lower bounds for the multi-pass streaming complexity of Dyck languages. We first plug Theorem 1 into a direct sum theorem, which lower bounds the communication cost of a problem we call MULTI-AI (for "multiple copies of AUGMENTED-INDEX"). We then reduce MULTI-AI to, e.g., DYCK(2). The direct sum theorem is a natural extension to multiple passes of a similar single-pass theorem ([16], where the relevant problems are called ASCENSION and MOUNTAIN). Thus, on the lower bound side, our chief contribution is Theorem 1, and its most important consequence is the *multi-pass* nature of the resulting lower bounds. This demonstrates a curious phenomenon: an explicit, natural data stream problem that is fairly easy given two passes in opposite directions (Magniez et al. give an $O(\log^2 N)$-space algorithm), which is exponentially harder if only multiple unidirectional passes are allowed.

A second application is to *memory checking*, introduced by Blum et al. [4] and continued by numerous groups including Ajtai [2], Chu et al. [7], Dwork et al. [11], and Naor and Rothblum [18]. The problem, as considered in this paper, is to observe a sequence of N updates and queries to (an implementation of) a data structure, and to report whether or not the implementation operated correctly on the instance observed. A concrete example is to observe a transcript of operations on a priority queue: we see a sequence of insertions intermixed with items claimed to be the results of extractions, and the problem is to decide whether this is correct. Much previous work allowed *invasive* checkers, which modify the inserted items and/or introducing additional read operations. However, when the checker is restricted to being completely *passive* (can only observe) we study the (streaming) complexity of *recognizing* valid transcripts. For instance, define PQ to be the language of valid transcripts of priority queue operations that start and end with an empty queue. One can similarly define languages STACK and DEQUE (for double-ended queues). The invasive protocols of [4] typically modified the input items by attaching a "timestamp" to each inserted item, which suggests variant languages PQ-TS, STACK-TS, and DEQUE-TS, where each extraction is augmented by the timestamp of its corresponding insertion. Though we briefly study these variant languages towards the end of this paper, we consider the languages without auxiliary information to be more natural and practical.

We present new algorithms for these memory checking languages: PQ, STACK, and DEQUE can each be recognized in $\tilde{O}(\sqrt{N})$ space and one pass. On the lower bound side, Theorem 1 and MULTI-AI give $\Omega(\sqrt{N})$ bounds for each of these problems, even allowing multiple passes over the transcript. Our upper bound for PQ strengthens the $\tilde{O}(\sqrt{N})$ bound of Chu et al. [7] for PQ-TS. This strengthening is significant, for timestamps can radically simplify problems: we note that STACK-TS can be recognized in just $O(\log N)$ space, in marked contrast to STACK.

Highlights: As we view Theorem 1 as our main technical contribution, we first give a careful exposition of its proof, in Section II. The main technical hurdle is dealing with the fact that Alice and Bob share some input, which breaks the useful "rectangle property." (reminiscent of number-on-the-forehead communication [6], where input sharing makes strong lower bounds rather hard to prove.) The highlight of our proof is the Fat Transcript Lemma (Lemma 7), with its careful interplay between a suitably weakened rectangle property (Lemma 6) and the information cost measure.

After discussion (Section III) of the direct sum theorem and its implications for MULTI-AI, we address language recognition, in Section IV. The highlight of this section is our algorithm to recognize PQ. Rather than determining whether the interaction sequence is valid directly, the algorithm conceptually reorders inserts and extracts so that the new sequence is valid iff the original sequence is. This reordering is designed such that the new sequence can easy be verified in small space.

Related Work: Independent of our work, Jain and Nayak recently presented an alternative proof for the lower bound result [12].

II. Augmented Index and Information Cost

Let $\text{AI} = \text{AI}_n$ (short for AUGMENTED-INDEX) denote the communication problem where Alice has a string $x \in \{0,1\}^n$, Bob has an index $k \in [n]$, the length-$(k-1)$ prefix of x, which we denote $x_{1:k-1}$, and a check bit $c \in \{0,1\}$. The goal is to output $\text{AI}(x,k,c) := x_k \oplus c$, i.e., to output 1 iff $x_k \neq c$.

We formalize the notion of information cost via the most general model of randomization in communication protocols: the parties may share a public coin, and separately, each party may have its own private coin. Let P be such a randomized protocol for AI, let ξ be a distribution on $\{0,1\}^n \times [n] \times \{0,1\}$ (effectively, a distribution on legal inputs to P) and let $(X,K,C) \sim \xi$. Let R denote the public random string used by P, and let T denote the transcript of messages sent by Alice and Bob (including the final output bit) in response to this random input (X,K,C): in general, T depends on X,K,C,R and the (unnamed) private random strings of the players. Define the *information cost* of P under ξ to be the pair of real numbers $(\text{icost}^A_\xi(P), \text{icost}^B_\xi(P))$:

$$\begin{aligned}\text{icost}^A_\xi(P) &:= \mathrm{I}(T : X \mid X_{1:K-1}, K, C, R)\\ \text{icost}^B_\xi(P) &:= \mathrm{I}(T : K, C \mid X, R)\,.\end{aligned}$$

Here, $\mathrm{I}(Y_1 : Y_2 \mid Y_3)$ is the conditional mutual information. These quantities reflect the amount of information that Alice reveals to Bob (and vice versa), through the transcript, about the portion of the input he does not see. In the above definition, conditioning on R is crucial, else it is simple to make these costs zero. From basic information theory, regardless of the choice of ξ, these costs are bounded from above by the number of bits communicated by Alice and Bob, respectively, in P. Thus, a tradeoff lower bound on information cost is a stronger than a similar tradeoff on bits communicated. We now turn to the choice of input distribution.

Definition 2. Let μ denote the uniform distribution on $\{0,1\}^n \times [n] \times \{0,1\}$. For $(X,K,C) \sim \mu$, let $\mu_0 := \mu \mid (X_K = C)$. Note that $\mathbb{E}_\mu[\text{AI}(X,K,C)] = \frac{1}{2}$, whereas $\mathbb{E}_{\mu_0}[\text{AI}(X,K,C)] = 0$. Thus, intuitively, μ is a hard distribution for AI, whereas μ_0 is an easy distribution.

The next technical lemma formalizes an averaging argument used in the main theorem. It follows by the Markov inequality and the union bound.

Lemma 3. *Consider functions $f_1,\ldots,f_L : D \to \mathbb{R}^+$, and $b_1,\ldots,b_L \in \mathbb{R}^+$, where $L > 0$ is an integer and D is a finite domain. Let Z be a random variable over D. Then*

$$\forall i \in [L]\ \mathbb{E}[f_i(Z)] \le b_i \ \Rightarrow\ \exists z \in D\ \forall i \in [L]\ f_i(z) \le L b_i\,.$$

Theorem 4 (Main Theorem; formal version of Theorem 1)**.** *There exists a constant ε such that if P is a randomized protocol for AI_n with error at most ε under μ, then either $\text{icost}^A_{\mu_0}(P) = \Omega(n)$ or $\text{icost}^B_{\mu_0}(P) = \Omega(1)$. In particular, the same tradeoff holds if P has worst case two-sided error at most ε.*

Proof: We split this proof into two parts. First, assuming the contrary, we zoom in on a specific setting of the public random string of P and a single transcript that has certain "fatness" properties that play a role analogous to the "large rectangles" seen in elementary communication complexity. This part of the proof is reminiscent of arguments in Pătraşcu's proof of the lopsided set disjointness lower bound [19]. Next, and more interestingly, we use these fatness properties to derive a contradiction, in Lemma 7. Throughout the proof, and the rest of this section, we tacitly assume that n is large enough.

Assume, to the contrary, that for every choice of constants ε, δ_1 and δ_2, there exists an ε-error protocol P^* for AI with $\text{icost}^A_{\mu_0}(P^*) \le \delta_1 n$ and $\text{icost}^B_{\mu_0}(P^*) \le \delta_2$. To write these conditions formally, let T^* denote the transcript of P^* (which uses a public random string R) on input $(X,K,C) \sim \mu$; we condition on $X_K = C$ when necessary, to effectively change the input distribution to μ_0. We adopt the convention that a transcript, t, also specifies its final output bit, $\text{out}(t)$. Then

$$\begin{aligned}\Pr[\text{out}(T^*) \neq \text{AI}(X,K,C)] &\le \varepsilon\,,\\ \mathrm{I}(T^* : X \mid X_{1:K-1}, K, C, R, X_K = C) &\le \delta_1 n\,,\\ \mathrm{I}(T^* : K, C \mid X, R, X_K = C) &\le \delta_2\,.\end{aligned}$$

These three inequalities can be interpreted as bounding the expectations of three non-negative functions of the random string R. Any particular setting of R reduces P^* to a private-coin protocol. Thus, applying Lemma 3 to these three inequalities, we see that there exists a *private-coin* protocol P for AI such that, if T denotes the transcript of P on input $(X,K,C) \sim \mu$, then

$$\Pr[\text{out}(T) \neq \text{AI}(X,K,C)] \le 3\varepsilon\,, \tag{1}$$

$$\mathrm{I}(T : X \mid X_{1:K-1}, K, C, X_K = C) \le 3\delta_1 n\,, \tag{2}$$

$$\mathrm{I}(T : K, C \mid X, X_K = C) \le 3\delta_2\,. \tag{3}$$

Straightforward calculation shows $\mathrm{H}(X \mid X_{1:K-1}, K, C, X_K = C) = (n-1)/2$ and that $\mathrm{H}(K,C \mid X, X_K = C) = \log n$. Thus, by the characterization of mutual information in terms of entropy, we can rewrite (2) and (3) as

$$\frac{n-1}{2} - \mathrm{H}(X \mid T, X_{1:K-1}, K, C, X_K = C) \le 3\delta_1 n\,, \tag{4}$$

$$\log n - \mathrm{H}(K, C \mid T, X, X_K = C) \le 3\delta_2\,. \tag{5}$$

Definition 5. Let ν denote the distribution of T and let $\nu_0 := \nu \mid (X_K = C)$. For a specific transcript t, let ρ_t denote the distribution $\mu \mid (T = t)$.

We can interpret (4) and (5) as bounding the expectations of appropriate functions of a random transcript distributed according to ν_0. Inequality (1), though, is not of this form, since there is no conditioning on $(X_K = C)$; instead, it says

$$\mathbb{E}_{T\sim\nu}\left[\Pr_{(X',K',C')\sim\rho_T}[\mathrm{out}(T) \neq \mathrm{AI}(X',K',C')]\right] \;\leq\; 3\varepsilon\,. \quad (6)$$

Since $\Pr[X_K = C] = \frac{1}{2}$, every transcript t satisfies $\nu_0(t) \leq 2\nu(t)$. Thus, switching the distribution in the outer expectation from ν to ν_0 can at most double the left-hand side. In other words,

$$\mathbb{E}_{T_0\sim\nu_0}\left[\Pr_{(X',K',C')\sim\rho_{T_0}}\left[\mathrm{out}(T_0) \neq \mathrm{AI}(X',K',C')\right]\right] \leq 6\varepsilon\,. \quad (7)$$

Finally, transcripts drawn from ν_0 typically output "0", as

$$\Pr_{T_0\sim\nu_0}[\mathrm{out}(T_0) \neq 0] = \Pr[\mathrm{out}(T) \neq \mathrm{AI}(X,K,C) \mid X_K = C] \leq 6\varepsilon\,, \quad (8)$$

where the final step uses (1). By another averaging argument, applying Lemma 3 to the four inequalities (4), (5), (7) and (8), we conclude that there exists a transcript t so

$$\begin{aligned}
\frac{n-1}{2} - \mathrm{H}(X \mid X_{1:K-1},K,C,X_K = C, T = t) \;&\leq\; 12\delta_1 n\,,\\
\log n - \mathrm{H}(K,C \mid X, X_K = C, T = t) \;&\leq\; 12\delta_2\,,\\
\Pr_{(X',K',C')\sim\rho_t}[\mathrm{out}(t) \neq \mathrm{AI}(X',K',C')] \;&\leq\; 24\varepsilon\,,\\
\mathrm{out}(t) \;&=\; 0\,.
\end{aligned}$$

However, by the Fat Transcript Lemma (Lemma 7) below, it follows that no transcript can simultaneously satisfy the above four conditions. This completes the proof. ∎

At this point, we need to understand what is special about the distributions ρ_t (from Definition 5), given that they arise from transcripts of private-coin communication protocols. The key fact needed here is the so-called *rectangle property* of deterministic communication protocols [15, Ch. 1], specifically, its extension to private-coin randomized protocols, as used, e.g., by Bar-Yossef et al. [3, Lemma 6.7].

The complication here is due to the fact that Alice and Bob share some information. Had Bob *not* received any part of Alice's input, ρ_t would have been a product of a distribution on values of x, and another distribution on values of (k,c). But because Bob *does*, in fact, start out knowing $x_{1:k-1}$, we can only draw the weaker conclusion given in the following lemma.

Lemma 6. *Let $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{(w,k,c) \in \{0,1\}^* \times [n] \times \{0,1\} : |w| = k-1\}$. Let P be a private-coin protocol in which Alice receives a string $x \in \mathcal{X}$ while Bob receives $(w,k,c) \in \mathcal{Y}$, with the promise that $w = x_{1:k-1}$. Then, for every transcript t of P, there exist functions $p_{A,t} : \mathcal{X} \to \mathbb{R}^+$ and $p_{B,t} : \mathcal{Y} \to \mathbb{R}^+$ such that $\forall (x,k,c) \in \{0,1\}^n \times [n] \times \{0,1\}$:*

$$\rho_t(x,k,c) \;=\; p_{A,t}(x)\cdot p_{B,t}(x_{1:k-1},k,c)\,.$$

Proof: Let $\mathcal{T}$ be the set of all possible transcripts of P and let T be a random transcript of P on input $(X,K,C) \sim \mu$. By the rectangle property for private-coin protocols (Lemma 6.7 of [3]), there exist mappings $q_A : \mathcal{T} \times \mathcal{X} \to \mathbb{R}^+$ and $q_B : \mathcal{T} \times \mathcal{Y} \to \mathbb{R}^+$ such that

$$\begin{aligned}
&\Pr[T = t \mid (X,K,C) = (x,k,c)]\\
&\quad=\; q_A(t;x)\cdot q_B(t;x_{1:k-1},k,c)\,.
\end{aligned}$$

Recall that μ is just a uniform distribution. In particular, it decomposes as $\mu(x,k,c) = \mu_A(x)\mu_B(k,c)$. Thus, by Bayes' Theorem,

$$\begin{aligned}
\rho_t(x,k,c) \;&=\; \frac{\mu(x,k,c)\cdot\Pr[T = t \mid (X,K,C) = (x,k,c)]}{\Pr[T = t]}\\
&=\; \frac{\mu_A(x)\cdot\mu_B(k,c)\cdot q_A(t;x)\cdot q_B(t;x_{1:k-1},k,c)}{\Pr[T = t]}\,.
\end{aligned}$$

Now set $p_{A,t}(x) := \mu_A(x)\cdot q_A(t;x)/\Pr[T = t]$ and $p_{B,t}(w,k,c) := \mu_B(k,c)\cdot q_B(t;w,k,c)$. ∎

We now state the promised lemma that finishes the proof of Theorem 4. We alert the reader that, *from here on, the distribution of (X,K,C) is no longer uniform*; instead, we condition the uniform distribution on a specific transcript.

Lemma 7 (Fat Transcript Lemma)**.** *There exist positive real constants ε_1, δ_3 and δ_4 such that, for every transcript t of a private-coin communication protocol for* AI*, with* $\mathrm{out}(t) = 0$*, we have the following. Let $(X,K,C) \sim \rho_t$. Then the following conditions do not hold simultaneously:*

$$\mathrm{H}(X \mid X_{1:K-1},K,C,X_K = C) \geq (1/2 - \delta_3)n\,, \quad (9)$$

$$\mathrm{H}(K,C \mid X, X_K = C) \geq \log n - \delta_4\,, \quad (10)$$

$$\mathbb{E}[\mathrm{AI}(X,K,C)] \leq \varepsilon_1\,. \quad (11)$$

Proof: Suppose, to the contrary, that (9), (10) and (11) do hold for every choice of ε_1, δ_3 and δ_4. Since C is determined by X and K whenever the condition $X_K = C$ holds, the left-hand side of (10) equals $\mathrm{H}(K \mid X, X_K = C)$. We now expand (11). In what follows, we use notation of the form "$u0v$" to denote the concatenation of the string u, the length-1 string "0", and the string v.

$$\begin{aligned}
&\mathbb{E}[\mathrm{AI}(X,K,C)] && (12)\\
&=\; \sum_{k=1}^{n}\sum_{x\in\{0,1\}^n}\sum_{c\in\{0,1\}} \rho_t(x,k,c)\cdot\mathrm{AI}(x,k,c)\\
&=\; \sum_{k=1}^{n}\sum_{\substack{u\in\{0,1\}^{k-1},\ b\in\{0,1\}\\ v\in\{0,1\}^{n-k},\ c\in\{0,1\}}} \rho_t(ubv,k,c)\cdot\mathrm{AI}(ubv,k,c)\,. && (13)
\end{aligned}$$

Let $p_A = p_{A,t}$ and $p_B = p_{B,t}$ be the functions given by Lemma 6. Let ρ' denote the distribution $\rho_t \mid (X_K = C)$ and λ denote the distribution of (X,K) conditioned on $(X_K = C)$, i.e., let $\lambda(x,k) = \rho'(x,k,0) + \rho'(x,k,1)$. Observe that for every triple (x,k,c), we have

$$\rho_t(x,k,c) \;\geq\; \Pr[X_K = C]\cdot\rho'(x,k,c) \;\geq\; (1-\varepsilon_1)\rho'(x,k,c)$$

where the last step uses (11). Noting $\textsc{ai}(ubv,k,c)=1$ iff $b\neq c$, we manipulate (13) using the inequality above at (14).

$$
\begin{aligned}
&\mathbb{E}[\textsc{ai}(X,K,C)]\\
&= \sum_{k=1}^{n}\sum_{u\in\{0,1\}^{k-1}}\sum_{v\in\{0,1\}^{n-k}}\Big(\rho_t(u0v,k,1)+\rho_t(u1v,k,0)\Big)\\
&= \sum_{\substack{1\le k\le n\\ u\in\{0,1\}^{k-1}}}\sum_{v\in\{0,1\}^{n-k}}\Big(p_A(u0v)\cdot p_B(u,k,1)+p_A(u1v)\cdot p_B(u,k,0)\Big)\\
&= \sum_{\substack{1\le k\le n\\ u\in\{0,1\}^{k-1}}}\left(\begin{array}{c} p_B(u,k,1)\sum_{v\in\{0,1\}^{n-k}}p_A(u0v)\\ +\ p_B(u,k,0)\sum_{v\in\{0,1\}^{n-k}}p_A(u1v)\end{array}\right)\\
&\ge \sum_{\substack{1\le k\le n\\ u\in\{0,1\}^{k-1}}}\Big(p_B(u,k,0)+p_B(u,k,1)\Big)\cdot\min\left\{\begin{array}{l}\sum_{v\in\{0,1\}^{n-k}}p_A(u0v),\\ \sum_{v\in\{0,1\}^{n-k}}p_A(u1v)\end{array}\right\}\\
&\ge (1-\varepsilon_1)\sum_{\substack{1\le k\le n\\ u\in\{0,1\}^{k-1}}}\min\Big\{\sum_{v\in\{0,1\}^{n-k}}\big(\rho'(u0v,k,0)+\rho'(u0v,k,1)\big),\\
&\qquad\qquad \sum_{v\in\{0,1\}^{n-k}}\big(\rho'(u1v,k,0)+\rho'(u1v,k,1)\big)\Big\} \qquad (14)\\
&= (1-\varepsilon_1)\sum_{\substack{1\le k\le n\\ u\in\{0,1\}^{k-1}}}\min\Big\{\sum_{v\in\{0,1\}^{n-k}}\lambda(u0v,k),\sum_{v\in\{0,1\}^{n-k}}\lambda(u1v,k)\Big\}. \qquad (15)
\end{aligned}
$$

Let $\alpha:\{0,1\}^n\to[0,1]$ and $\beta:[n]\to[0,1]$ be the marginals of λ, i.e., $\alpha(x):=\sum_{k=1}^n\lambda(x,k)$ and $\beta(k):=\sum_{x\in\{0,1\}^n}\lambda(x,k)$. Crucially, for these distributions,

Claim 8. *We have* $\|\lambda-\alpha\otimes\beta\|_1=\sum_{x\in\{0,1\}^n}\sum_{k=1}^n|\lambda(x,k)-\alpha(x)\beta(k)|\le\sqrt{(2\ln 2)\cdot\delta_4}$.

Proof: Using the characterization of mutual information in terms of Kullback-Leibler divergence, we get

$$
\begin{aligned}
&D_{KL}(\lambda\parallel\alpha\otimes\beta)\\
&= \mathrm{I}(K:X\mid X_K=C)\\
&= \mathrm{H}(K\mid X_K=C)-\mathrm{H}(K\mid X,X_K=C)\ \le\ \delta_4,
\end{aligned}
$$

where the last step uses (10) and the basic fact that $\mathrm{H}(K\mid X_K=C)\le\log n$. The claim now follows from Pinsker's inequality (for which see, e.g., [9, Lemma 12.6.1]). ∎

Claim 9. *We have* $\sum_{k=1}^n|\beta(k)-1/n|\le\sqrt{(2\ln 2)\cdot\delta_4}$.

Proof: Relax (10) to $\mathrm{H}(K\mid X_K=C)\ge\log n-\delta_4$. Let γ be the uniform distribution on $[n]$. Then $D_{KL}(\beta\parallel\gamma)=\log n-\mathrm{H}(K\mid X_K=C)\le\delta_4$. Now apply Pinsker's inequality. ∎

Let $\delta_5:=\sqrt{(2\ln 2)\cdot\delta_4}$. Using Claim 8 to estimate the expression (15), keeping in mind that any particular $\lambda(x,k)$ term appears at most once in the summation, we get

$$
\begin{aligned}
&\frac{\mathbb{E}[\textsc{ai}(X,K,C)]}{1-\varepsilon_1}\\
&\ge \sum_{\substack{1\le k\le n\\ u\in\{0,1\}^{k-1}}}\beta(k)\min\left\{\sum_{v\in\{0,1\}^{n-k}}\alpha(u0v),\sum_{v\in\{0,1\}^{n-k}}\alpha(u1v)\right\}-\delta_5. \qquad (16)
\end{aligned}
$$

For each $k\in[n]$, define the distribution $\hat{\alpha}_k$ on $\{0,1\}^{k-1}$ by $\hat{\alpha}_k(u):=\sum_{w\in\{0,1\}^{n-k+1}}\alpha(uw)=\Pr[X_{1:k-1}=u\mid X_K=C]$.

Let $\mathrm{H}_b:[0,1]\to[0,1]$ be the binary entropy function, i.e., $\mathrm{H}_b(z):=-z\log z-(1-z)\log(1-z)$, and $\mathrm{H}_b^{-1}:[0,1]\to[0,\frac{1}{2}]$ its (well-defined) inverse. Note that if Z is a binary random variable, then $\min\{\Pr[Z=0],\Pr[Z=1]\}=\mathrm{H}_b^{-1}(\mathrm{H}(Z))$. Thus,

$$
\begin{aligned}
&\min\left\{\sum_{v\in\{0,1\}^{n-k}}\alpha(u0v),\sum_{v\in\{0,1\}^{n-k}}\alpha(u1v)\right\} \qquad (17)\\
&\quad= \hat{\alpha}_k(u)\cdot\min\left\{\frac{\hat{\alpha}_{k+1}(u0)}{\hat{\alpha}_k(u)},\frac{\hat{\alpha}_{k+1}(u1)}{\hat{\alpha}_k(u)}\right\}\\
&\quad= \hat{\alpha}_k(u)\cdot\mathrm{H}_b^{-1}\Big(\mathrm{H}(X_k\mid X_{1:k-1}=u,X_K=C)\Big). \qquad (18)
\end{aligned}
$$

Plugging this back into (16), we obtain

$$
\begin{aligned}
&\frac{\mathbb{E}[\textsc{ai}(X,K,C)]}{1-\varepsilon_1}+\delta_5\\
&\ge \sum_{k=1}^n\beta(k)\sum_{u\in\{0,1\}^{k-1}}\hat{\alpha}_k(u)\cdot\mathrm{H}_b^{-1}\Big(\mathrm{H}(X_k\mid X_{1:k-1}=u,X_K=C)\Big)\\
&\ge \sum_{k=1}^n\beta(k)\cdot\mathrm{H}_b^{-1}\sum_{u\in\{0,1\}^{k-1}}\hat{\alpha}_k(u)\cdot\mathrm{H}(X_k\mid X_{1:k-1}=u,X_K=C) \qquad (19)\\
&= \sum_{k=1}^n\beta(k)\cdot\mathrm{H}_b^{-1}\Big(\mathrm{H}(X_k\mid X_{1:k-1},X_K=C)\Big)\\
&\ge \mathrm{H}_b^{-1}\left(\sum_{k=1}^n\beta(k)\cdot\mathrm{H}(X_k\mid X_{1:k-1},X_K=C)\right) \qquad (20)\\
&\ge \mathrm{H}_b^{-1}\left(\sum_{k=1}^n\frac{1}{n}\cdot\mathrm{H}(X_k\mid X_{1:k-1},X_K=C)-\delta_5\right) \qquad (21)\\
&= \mathrm{H}_b^{-1}\left(\frac{\mathrm{H}(X\mid X_K=C)}{n}-\delta_5\right), \qquad (22)
\end{aligned}
$$

where (19) and (20) follow from Jensen's inequality, using the convexity of H_b^{-1}, (21) uses Claim 9 and the fact that H_b^{-1} is increasing on $[0,1]$, and (22) uses the chain rule for entropy. By relaxing (9), we have $\mathrm{H}(X\mid X_K=C)\ge(1/2-\delta_3)n$. Using this and (11), we now obtain

$$\frac{\varepsilon_1}{1-\varepsilon_1}+\delta_5\ \ge\ \mathrm{H}_b^{-1}(1/2-\delta_3-\delta_5).$$

Recall that $\delta_5=\sqrt{(2\ln 2)\cdot\delta_4}$. By choosing ε_1, δ_3 and δ_4 small enough, we can make the left-hand side of the above inequality approach zero and the right-hand side approach $\mathrm{H}_b^{-1}(1/2)>0$, and we finally have our contradiction. ∎

III. A Direct Sum Argument

$\textsc{multi-ai}_{m,n}$ is a communication problem with $2m$ players $A_1,B_1,\ldots,A_m,B_m$. Each A_i receives a string $x^i\in\{0,1\}^n$ and each B_i receives an integer $k^i\in[n]$, a bit $c^i\in\{0,1\}$, and the length-(k^i-1) prefix $x^i_{1:k^i-1}$ of x^i. The players wish to compute the predicate $\bigvee_{i=1}^m\textsc{ai}_n(x^i,k^i,c^i)$. The players may use private random strings and a common public random string over p *rounds*. Each round has the players sending s bit messages privately to the next in the sequence:

$$A_1 \to B_1 \to A_2 \to B_2 \cdots \to A_m \to B_m \to A_m \to A_{m-1} \cdots \to A_1 .$$

At the end of p rounds, A_1 must announce the answer, which is required to be correct with probability at least $(1-\varepsilon)$ on any possible input. Call such a protocol a $[p,s,\varepsilon]$-protocol.

Theorem 10. *Every $[p,s,\frac{1}{3}]$-protocol for* $\text{MULTI-AI}_{m,n}$ *satisfies $ps = \Omega(\min\{m,n\})$.*

This theorem is easily seen to be near-optimal: even with $p=1$, we have a trivial protocol achieving $s = O(n)$ and another trivial protocol achieving $s = O(m \log n)$.

Notice that the augmented index problem studied in Section II satisfies $\text{AI}_n = \text{MULTI-AI}_{1,n}$. Intuitively, a protocol for $\text{MULTI-AI}_{m,n}$ must solve m independent AI instances, and thus, must use about m times the communication that a single instance requires. To prove Theorem 10, we formalize this intuition as a direct sum theorem, proved using a suitable refinement of the information complexity paradigm [5]. To state this direct sum theorem, we need a suitable notion of information cost for protocols solving MULTI-AI. Let Q be a $[p,s,\varepsilon]$-protocol for $\text{MULTI-AI}_{m,n}$. Let ξ be a distribution on inputs to Q and let M_m denote the sequence of messages sent by player B_m when Q is run on a random input $\langle (X^i,K^i,C^i)\rangle_{i=1}^m \sim \xi$, using a public random string R. We strategically define the *information cost* of Q under ξ as

$$\text{icost}_\xi(Q) \ := \ \text{I}(M_m : K^1, C^1, \ldots, K^m, C^m \mid X^1, \ldots, X^m, R) . \tag{23}$$

It is worth noting that when $m=1$, i.e., we are considering a protocol for AI, this definition specializes to that of $\text{icost}^B_\xi(Q)$ in (1). This is proved in Lemma 24 (in the appendix).

Theorem 11 (Direct sum theorem for AI)**.** *Suppose there exists a $[p,s,\varepsilon]$-protocol Q for* $\text{MULTI-AI}_{m,n}$*. Then there exists an ε-error randomized protocol P for* AI *in which Alice sends at most ps bits in total, such that $m \cdot \text{icost}^B_{\mu_0}(P) \ \le \ \text{icost}_{\mu_0^{\otimes m}}(Q)$, where μ_0 is as in Definition 2 and $\mu_0^{\otimes m}$ denotes the m-fold product of μ_0 with itself.*

This is a straightforward generalization, to multiple rounds, of a similar theorem of Magniez et al. [16], which applied only to restricted families of one-round protocols. Details appear in the Appendix. We can now prove our multi-round communication lower bound on MULTI-AI.

Proof of Theorem 10: Let ε be small enough for Theorem 4 to apply. By a standard error reduction argument, we may assume that we have a $[p,s,\varepsilon]$-protocol, Q, for $\text{MULTI-AI}_{m,n}$. From basic information theory, it follows that $\text{icost}_{\mu_0^{\otimes m}}(Q) \le ps$. By Theorem 11, there is an ε-error protocol P for AI with $\text{icost}^B_{\mu_0}(P) \le ps/m$ and in which Alice communicates at most ps bits, so that $\text{icost}^A_{\mu_0}(P) \le ps$. By Theorem 4, either $ps/m = \Omega(1)$ or $ps = \Omega(n)$; i.e., $ps = \Omega(\min\{m,n\})$. ■

IV. Streaming Language Recognition and Passive Memory Checking

In this section we present our results for recognizing certain languages in the data stream model. Of particular interest is $\text{DYCK}(2)$, the language of strings of well-balanced parentheses in two types of parentheses. Formally, representing '(', ')', '[', and ']' as a, $\bar{a}$, b, and $\bar{b}$ respectively,

Definition 12. $\text{DYCK}(2)$ is the language generated by the context-free grammar $S \to aS\bar{a} \mid bS\bar{b} \mid SS \mid \varepsilon$.

An important class of memory checking problems, which we call *passive checking*, can also be viewed as language recognition problems in the data stream model. For example, we define PQ to be the language corresponding to transcripts of operations, or "interaction sequences," of a priority queue that begins and ends with an empty queue. (Without this restriction, the resulting language would require $\Omega(N)$ space to recognize, for simple reasons [7, Theorem 4].) Formally,

Definition 13. An *interaction sequence* $\sigma = \sigma_1\sigma_2\ldots\sigma_N$ is a string over the alphabet $\Sigma = \{\text{ins}(u), \text{ext}(u) : u \in [U]\}$. Let $\text{PQ} = \text{PQ}(U)$ be the language defined over Σ where $\text{ins}(u)$ is interpreted as an insertion of u into a priority queue, and $\text{ext}(u)$ as an extraction of u from the priority queue. The state of the queue at step j is a multiset M_j where $M_0 = \emptyset$ and

$$M_j \ = \ \begin{cases} M_{j-1} \setminus \{\min(M_{j-1})\} & \text{if } \sigma_j = \text{ext}(v) \\ M_j \ = \ M_{j-1} \cup \{u\} & \text{if } \sigma_j = \text{ins}(u) \end{cases} . \tag{24}$$

Then $\sigma \in \text{PQ}$ for $|\sigma| = N$ iff $M_N = \emptyset$ and $\forall j \in [N]$ $(\sigma_j = \text{ext}(u) \Rightarrow u = \min(M_{j-1}))$.

A recognizer for PQ can also recognize $\text{DYCK}(2)$ via an online transformation. For a string p over parentheses $\{a,\bar{a},b,\bar{b}\}$ define $\text{height}(\varepsilon) = 0$, and

$$\text{height}(p) := |\{j : p_j \in \{a,b\}\}| - |\{j : p_j \in \{\bar{a},\bar{b}\}\}|$$

Transform p into $\psi(p) = \phi(p_{1:1})\phi(p_{1:2})\ldots\phi(p_{1:N})$ where:

$$\phi(p_{1:i}) = \begin{cases} \text{ins}(2N - 2\,\text{height}(p_{1:i-1})) & \text{if } p_i = a \\ \text{ext}(2N - 2\,\text{height}(p_{1:i})) & \text{if } p_i = \bar{a} \\ \text{ins}(2N - 2\,\text{height}(p_{1:i-1}) - 1) & \text{if } p_i = b \\ \text{ext}(2N - 2\,\text{height}(p_{1:i}) - 1) & \text{if } p_i = \bar{b} \end{cases}$$

For example, the string $\langle a,a,\bar{a},b,\bar{b},\bar{a}\rangle$ is transformed into $\langle \text{ins}(12), \text{ins}(10), \text{ext}(10), \text{ins}(9), \text{ext}(9), \text{ext}(12)\rangle$. We omit the full proof that $\psi(p) \in \text{PQ} \iff p \in \text{DYCK}(2)$ for brevity.

Lemma 14. *There exists an $O(\log N)$-space stream reduction from* $\text{DYCK}(2)$ *to* $\text{PQ}(4N)$.

We use Theorem 10 to resolve the conjecture of Magniez et al. [16] on the multi-pass complexity of $\text{DYCK}(2)$ and PQ.

Theorem 15 (Multi-pass Lower Bounds for DYCK and PQ)**.** *Let L denote either* $\text{DYCK}(2)$ *or* $\text{PQ}(N)$*. Suppose there exists* *a $\frac{1}{3}$-error, p-pass, s-space, randomized streaming algorithm that recognizes L on length-N streams. Then $ps = \Omega(\sqrt{N})$.*

Proof: Using the reduction of Magniez et al. [16], an ε-error p-pass randomized streaming algorithm for DYCK(2) that uses s bits of space on streams of length $\Theta(mn)$ can be turned into a $[p,s,\varepsilon]$-protocol for MULTI-AI$_{m,n}$. One can similarly reduce MULTI-AI$_{m,n}$ to PQ(N); this was implicitly claimed without proof in [16]. Alternatively, Lemma 14 gives an explicit reduction from MULTI-AI$_{m,n}$ to PQ via DYCK(2). To complete the proof, we combine these reductions with Theorem 10, setting $m = n$. ■

Unidirectional versus Bidirectional Passes: DYCK(2) can be recognized in $O(\log^2 N)$ space using two passes, one in each direction [16]. On the other hand, the above theorem implies that achieving polylog(n) space with only unidirectional access to the input would require $\widetilde{\Omega}(\sqrt{N})$ passes. To the best of our knowledge, this is the first explicit demonstration of such a strong separation.

A. Passive Checking of Priority Queues

Given the connection between PQ and DYCK(2) shown in Lemma 14, one might hope to adapt the algorithms of [16] to this problem. However, there seems to be no such easy reduction in this direction. For intuition, observe that DYCK(2) has a much stricter requirement on the permitted strings: if its second half consists of close-parentheses only, then its first half is uniquely determined. On the other hand, in PQ, one can find $(N/2)!$ sequences consisting of $N/2$ insertions followed by $N/2$ extractions that all agree on the second half, suggesting the two languages are quite different.

Theorem 16. *We can recognize the language* PQ *in one pass, using $O(\sqrt{N}(\log U + \log N))$ bits of space: an input $\sigma \in$ PQ is accepted with certainty, and an input $\sigma \notin$ PQ is rejected with probability $\geq 1 - 1/N^2$.*

Overview of the Algorithm: We first present a $O(Ur(\log U + \log N))$ space algorithm for the case when the input string can be decomposed as $\sigma = \sigma^{E_1}\sigma^{I_1}\sigma^{E_2}\sigma^{I_2}\ldots\sigma^{E_r}\sigma^{I_r}$ where σ^{E_i} is a sequence of extracts and σ^{I_i} is a sequence of inserts. We refer to $\sigma^{E_i}\sigma^{I_i}$ as the ith *epoch* of the string and note that, for sufficiently large r, any σ is of this form. After presenting the full space algorithm, we show how to transform σ such that $r = O(\sqrt{N})$ and subsequently, to reduce the space to $\tilde{O}(\sqrt{N})$. Last, it is easy to verify that the extracts in each σ^{E_i} are in ascending order and $\sigma^{E_1} = \sigma^{I_r} = \emptyset$, so we assume that both are true.

We present the algorithm PQ-CHECK as Algorithm 1.

1. For each epoch k, PQ-CHECK maintains $f[k]$, the maximum value that has been extracted after the kth epoch. In particular, at the start of the ith epoch, $f[i-1] = 0$.
2. Each insert/extract of u is *assigned* to the earliest epoch "consistent" with the current f values maintained by PQ-CHECK, i.e., $\ell = \min\{k : f[k] \leq u\}$. Each $\text{ext}(u) \in \sigma^{E_i}$ is assigned to an epoch between 1 and $i-1$ (this follows because the extracts in σ^{E_i} are in increasing order and $f[i-1]$ equals 0 when the first extract in σ^{E_i} is processed), while each $\text{ins}(u) \in \sigma^{I_i}$ is assigned to an epoch between 1 and i. Importantly, for $\sigma \in$ PQ, each $\text{ext}(u)$ will be assigned to the same epoch as the most recent $\text{ins}(u)$.
3. The algorithm maintains arrays X, Y, and Z to track information about occurrences of item u assigned to epoch k (we later use hashing techniques to reduce the size of this information). Informally, X tracks the number of insertions of u assigned to epoch k before the first extraction of u that is assigned to epoch k, while Y tracks the number of extractions of u assigned to epoch k minus the number of insertions of u assigned to epoch k from the first extraction of u assigned to epoch k onwards. A necessary condition is that these two counts should agree. However, this counting alone fails to detect extractions of u that appear before the corresponding insertions. Therefore, Z is used to identify the maximum "balance" of u during epoch k. This should also match X if the sequence is correct, and we later show that these are sufficient conditions to check membership in PQ.

Algorithm 1 PQ-CHECK

```
1:  input σ = σ^{E_1} σ^{I_1} σ^{E_2} σ^{I_2} ... σ^{E_r} σ^{I_r} where σ^{E_1} = σ^{I_r} = ∅
2:  for k ∈ {1,...,r}, u ∈ {1,...,U}, do f[k] ← 0, X[k,u] ←
    0, Y[k,u] ← 0, Z[k,u] ← 0
3:  for i ∈ {1,...,r} do
4:      for ext(u) ∈ σ^{E_i} do
5:          ℓ ← min{k : f[k] ≤ u}
6:          Y[ℓ,u] ← Y[ℓ,u] + 1
7:          Z[ℓ,u] ← max(Y[ℓ,u], Z[ℓ,u])
8:          for 1 ≤ k < i do f[k] ← max(u, f[k])
9:      end for
10:     for ins(u) ∈ σ^{I_i} do
11:         ℓ ← min{k : f[k] ≤ u}
12:         if f[ℓ] < u then X[ℓ,u] ← X[ℓ,u] + 1
13:         if f[ℓ] = u then Y[ℓ,u] ← Y[ℓ,u] − 1
14:     end for
15: end for
16: if X ≠ Z or X ≠ Y then reject else accept
```

Define $f_t(k) = \max\{u : \sigma_i = \text{ext}(u), |\sigma^{E_1}\ldots\sigma^{I_k}| + 1 \leq i \leq t\}$. For $u \in [U]$ and $t \in [N]$, define $b(t,u) = \min\{k : f_t(k) \leq u\}$. Given an interaction sequence σ and $u \in [U]$, define

$$\text{cnt}(\sigma, u) := |\{t : \sigma_t = \text{ins}(u)\}| - |\{t : \sigma_t = \text{ext}(u)\}| \ .$$

Lemma 17. *After processing the tth element, Algorithm 1 has computed $f[k] = f_t(k)$, i.e., the maximum value extracted after the end of the kth epoch. For all k, $f[k]$ is non-decreasing as t increases.*

Proof: Observe that Algorithm 1 only updates $f[k]$ in Line 8, for $k < i$ where the current epoch is the ith epoch.

The equivalence of $f[k]$ and $f_t(k)$ follows immediately by an inductive argument over t. $f[k] = f_t(k)$ is seen to be non-decreasing by inspection of the definition of $f_t(k)$. ■

Lemma 18. *Let $X_t(k,u), Y_t(k,u)$ and $Z_t(k,u)$ denote the values of $X[k,u], Y[k,u]$, and $Z[k,u]$ after processing the tth element. Assume that the first t elements of the interaction sequence are a prefix of some interaction sequence in* PQ, *i.e., for all $j \in [t], (\sigma_j = \text{ext}(v) \implies v = \min(M_{j-1}))$ where $\{M_j\}_{j=0}^{N}$ is the family of multisets defined in Eq.* (24)*. Then, for any $u \in [U]$ and $k = b(t,u)$, we have:*

$$\text{cnt}(\sigma_{1:t}, u) = X_t(k,u) - Y_t(k,u)$$

and for $k < b(t,u)$, $X_t(k,u) = Y_t(k,u)$.

Theorem 19. *If $\sigma \notin$ PQ, Algorithm 1 rejects, else it accepts.*

Proof: If $\sigma \notin$ PQ, consider the minimum t such that $\sigma_t = \text{ext}(u)$ and $u \neq \min(M_{t-1})$. Let $k = b(t-1,u)$. There are two possibilities. First, suppose $u \notin M_{t-1}$. Then, by Lemma 18, before processing σ_t, $X_{t-1}(k,u) - Y_{t-1}(k,u) = 0$. After processing σ_t we have $Y_t(k,u) = Y_{t-1}(k,u) + 1$. Hence,

$$Z_t(k,u) \;\geq\; Y_t(k,u) \;>\; X_t(k,u)\,.$$

Since $Z_s(k,u)$ is non-decreasing in s and $X_s(k,u) = X_t(k,u)$ for $s > t$ after $f(k)$ becomes equal to u, at the end of the algorithm $Z_N(k,u) \neq X_N(k,u)$. Hence the algorithm rejects σ. Otherwise, suppose $u \in M_{t-1}$ but $\min(M_{t-1}) = v \neq u$. Then $\text{cnt}(\sigma_{1:t-1}, v) > 0$. Let $k = b(t-1,v)$ and by Lemma 18, $X_{t-1}(k,v) - Y_{t-1}(k,v) > 0$. Once $\text{ext}(u)$ is processed, $f[k]$ is increased to u and hence $X_s(k,v) > Y_s(k,v)$ for all $s > t$, and the algorithm rejects.

If $\sigma \in$ PQ, then by Lemma 18, at $t = N$, $X_t(k,u) - Y_t(k,u) = 0$ for all u,k. Consequently, $Z_t(k,u) \geq Y_t(k,u) = X_t(k,u)$ for all k,u. Since $\text{cnt}(\sigma_t, u) \geq 0$ for any $\sigma \in$ PQ, $Y_t(k,u) \leq X_t(k,u)$ for all t. Hence $Z_t(k,u) \leq X_t(k,u)$ and so $X_N = Y_N = Z_N$ and the algorithm accepts. ■

Local Consistency: We now consider a substring σ' of σ and show that if it does not violate some local conditions, then w.l.o.g. it can be assumed to be in a specific form.

Definition 20. We say σ' is *locally consistent* if both

1) $\forall i < k, u < v : (\sigma'_i = \text{ins}(u)) \wedge (\sigma'_k = \text{ext}(v)) \implies (\text{cnt}(\sigma'_{i+1:k-1}, u) < 0)$.
2) $\forall i < k, u > v : (\sigma'_i = \text{ext}(u)) \wedge (\sigma'_k = \text{ext}(v)) \implies (\text{cnt}(\sigma'_{i+1:k-1}, v) > 0)$.

Observe that if σ' is not locally consistent, then $\sigma \notin$ PQ, since the identified subsequence includes an extraction of an item which cannot be the smallest in the priority queue.

Lemma 21. *Given $\sigma = \sigma^{\text{pref}}\sigma'\sigma^{\text{suff}}$. If σ' is locally consistent, then there exists a mapping $\gamma(\sigma') = \sigma^a\sigma^b\sigma^c\sigma^d$ such that $\sigma^{\text{pref}}\sigma'\sigma^{\text{suff}} \in$ PQ iff $\sigma^{\text{pref}}\gamma(\sigma')\sigma^{\text{suff}} \in$ PQ. Here, σ^a and σ^c are both sequences of extracts in increasing order; and σ^b and σ^d are both sequences of inserts. There exists an algorithm* SUB-CHECK *tests if σ' is locally consistent and, if so, computes $\gamma(\sigma')$ in time $O(|\sigma'|\log|\sigma'|)$.*

Algorithm 2 SUB-CHECK

input σ'
$f \leftarrow 0$; $w \leftarrow 0$; $E \leftarrow \emptyset$; $I \leftarrow \{\infty\}$
for $i \in [|\sigma'|]$ **do**
  if $\sigma'_i = \text{ins}(u)$ **then** $I \leftarrow I \cup \{u\}$
  if $\sigma'_i = \text{ext}(v)$ **then**
    $m \leftarrow \min(I)$
    if $(v > m)$ **then** reject
    if $(v = m)$ **then** $I \leftarrow I \setminus \{v\}$; $w \leftarrow \max(w,v)$
    if $(v < m)$ **then**
      if $v < \max(f,w)$ **then** reject
      $f \leftarrow v$; $E \leftarrow E \cup \{v\}$
    end if
  end if
end for
output $\langle \text{ext}(v_1)\ldots\text{ext}(v_{|E|})\,\text{ins}(w)\,\text{ext}(w)\,\text{ins}(u_1)\ldots\text{ins}(u_{|I|})\rangle$ where v_i and u_i are the ith smallest values of E and I

We defer the proof and description of SUB-CHECK to the full version. Consequently, by breaking σ into sequential substrings of length l and reordering each substring (unless we determine the substring is not locally consistent) we may ensure that the interaction sequence has the form $\sigma = \sigma^{E_1}\sigma^{I_1}\sigma^{E_2}\sigma^{I_2}\ldots\sigma^{E_r}\sigma^{I_r}$ where $r = 2\lceil N/l\rceil$. The final algorithm runs PQ-CHECK and SUB-CHECK in parallel. The space required by SUB-CHECK is $O(l\log U)$ bits and PQ-CHECK can be implemented in $O(r(\log N + \log U))$ bits. Setting $l = \sqrt{N}$ yields Theorem 16.

Small-Space Implementation: Rather than maintain the arrays X, Y, and Z explicitly in PQ-CHECK, it suffices to keep a linear hash (which serves as a homomorphic fingerprint) of each array. These fingerprints can be compared, and if they match in Line 16, then, with high probability, the arrays agree. In Line 7 we need to perform a max operation between two values. This can be done by maintaining $Y[k, f_t(k)]$ and $Z[k, f_t(k)]$ explicitly for each k. At any time, there are at most r such values that are needed: observe that when $f_{t+1}(k) > f_t(k)$, $Y[k, f_t(k)]$ and $Z[k, f_t(k)]$ are never subsequently altered. The new values for $Y[k, f_{t+1}(k)]$ and $Z[k, f_{t+1}(k)]$ are initialized to 0. Hence, the space of the algorithm is $O(r)$ words to store the $Y[k, f[k]]$, $Z[k, f[k]]$ and $f[k]$ values, and a constant number of fingerprints to represent X, Y, and Z.

B. Passive Checking of Stacks, Queues, and Deques

Stack: Let STACK denote the language over interaction sequences that corresponds to stack operations. Now $\text{ins}(u)$ corresponds to an insertion of u to a stack, and $\text{ext}(u)$ is an extraction of u from the stack. Then $\sigma \in$ STACK iff σ corresponds to a valid transcript of operations on a stack which starts and ends empty. That is, the state of the stack at any step j can be represented by a string S^j so that $S^0 = \emptyset$, $S^j = uS^{j-1}$ if $\sigma_j = \text{ins}(u)$ and $S^j = S^{j-1}_{2:|S^{j-1}|}$ if $\sigma_j = \text{ext}(u)$. Then $\sigma \in$ STACK for $|\sigma| = N$ iff

$$S^N = \emptyset \quad \text{and} \quad \forall j \in [N], (\sigma_j = \text{ext}(u) \implies u = S^{j-1}_1)$$

Theorem 22. *Every $\frac{1}{3}$-error, p-pass, s-space randomized streaming algorithm to recognize* STACK *on length N streams must satisfy $ps = \Omega(\sqrt{N})$.* STACK *can be recognized in one pass with $O(\sqrt{N}\log N)$ bits of space with high probability.*

Proof: First, we observe that for $U = 2$, DYCK(U) = STACK if we associate $\text{ins}(u)$ with u and $\text{ext}(u)$ with $\bar{u}$. Therefore, the lower bound follows immediately. For the upper bound, the one-pass algorithm from [16] to recognize DYCK(2) can also recognize STACK over arbitrary U by appealing to their reduction from DYCK(U) to DYCK(2) or noting that their algorithm is directly applicable. ■

Queue: Let QUEUE denote the language over interaction sequences that correspond to queues. That is, the state of the queue at any step j can be represented by a string Q^j so that $Q^0 = \emptyset$, $Q^j = Q^{j-1}u$ if $\sigma_j = \text{ins}(u)$ and $Q^j = Q^{j-1}_{2:|Q^{j-1}|}$ if $\sigma_j = \text{ext}(u)$. Then $\sigma \in$ QUEUE for $|\sigma| = N$ iff

$$Q^N = \emptyset \quad \text{and} \quad \forall j \in [N], (\sigma_j = \text{ext}(u) \implies u = Q^{j-1}_1)$$

As observed in [4], QUEUE can be recognized with a single pass and $O(\log N)$ space: a single fingerprint checks that the value of the ith insert equals the value of the ith extract for all $i \in [N]$.

Deque: Let DEQUE denote the language over interaction sequences that corresponds to double-ended queues, i.e. there are two types of insert and extract operations, one operation for the head and one for the tail. Clearly, since a deque can simulate a stack via operations on the tail only, recognizing DEQUE is at least as hard as recognizing STACK. For the upper bound, it is possible to adapt the algorithm of [16]. Again, each block of $\sqrt{N}$ operations is partitioned into a prefix of extractions (to head and tail) and insertions (to head and tail). Now we maintain a deque of hash values of item, height pairs. Each extract to the head is applied to the hash at the head of the deque of hashes, and each extract to the tail is applied to the hash at the tail of the deque. The same check is applied: any hash which should now summarize no items must be identically zero (else, the algorithm rejects). Inserts to the head are parceled up into a hash which is placed at the head of the deque, and inserts to the tail are placed in a hash at the tail of the deque. Then we accept σ if after processing σ the algorithm reaches an empty deque and has not rejected at any point. Thus,

Theorem 23. *Every $\frac{1}{3}$-error, p-pass, s-space randomized streaming algorithm to recognize* DEQUE *on length N streams must satisfy $ps = \Omega(\sqrt{N})$.* DEQUE *can be recognized in one pass with $O(\sqrt{N}\log N)$ bits of space with high probability.*

C. Variations with timestamps

As noted in the introduction, the results of Blum et al. [4] can be viewed as recognizing languages where each $\text{ext}(u)$ is augmented with the timestamp of its matching $\text{ins}(u)$, and is denoted $\text{ext}(u,t)$. These languages are defined as before, but with the additional constraints that each $t \in [N]$ appears at most once across all extracts and

$$\forall j \in [N], \ (\sigma_j = \text{ext}(v,t) \implies \sigma_t = \text{ins}(v))$$

This defines the variant languages QUEUE-TS, STACK-TS, DEQUE-TS and PQ-TS. The observations of Blum et al. imply that verifying strings in STACK-TS and QUEUE-TS (and ensuring that all the timestamps are also consistent) requires only $O(\log N)$ space. The same argument also gives an $O(\log N)$ bound for deques. For PQ-TS, the problem seems harder: Chu et al. [7] gave an $\widetilde{O}(\sqrt{N})$ streaming algorithm relying heavily on the timestamps (and hence does not recognize PQ without timestamps). The complexity of PQ-TS remains open since the lower bound via augmented indexing does not apply.

REFERENCES

[1] F. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 157(2):139–159, 1996.

[2] M. Ajtai. The invasiveness of off-line memory checking. In *STOC*, 2002.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comp. Syst. Sci.*, 68(4):702–732, 2004.

[4] M. Blum, W. S. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[5] A. Chakrabarti, Y. Shi, A. Wirth, and A. C. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *FOCS*, 2001.

[6] A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *STOC*, 1983.

[7] M. Chu, S. Kannan, and A. McGregor. Checking and spot-checking of heaps. In *ICALP*, 2007.

[8] K. L. Clarkson and D. P. Woodruff. *Numerical linear algebra in the streaming model*. In *STOC*, 2009.

[9] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[10] K. Do Ba, P. Indyk, E. Price, and D. P. Woodruff. Lower bounds for sparse recovery. In *SODA*, 2010.

[11] C. Dwork, M. Naor, G. N. Rothblum, and V. Vaikuntanathan. How efficient can memory checking be? In *TCC*, 2009.

[12] R. Jain and A. Nayak. The space complexity of recognizing well-parenthesized expressions. in *Electronic Colloquium on Computational Complexity*, 071, 2010.

[13] R. Jain, J. Radhakrishnan, and P. Sen. A property of quantum relative entropy with an application to privacy in quantum communication. *J. ACM*, 56(6), 2009.

[14] D. M. Kane, J. Nelson, and D. P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *SODA*, 2010.
[15] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.
[16] F. Magniez, C. Mathieu, and A. Nayak. Recognizing well-parenthesized expressions in the streaming model. In *STOC*, 2010.
[17] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *J. Comp. Syst. Sci.*, 57(1):37–49, 1998.
[18] M. Naor and G. N. Rothblum. The complexity of online memory checking. *J. ACM*, 56(1), 2009.
[19] M. Pătraşcu. Unifying the landscape of cell-probe lower bounds. http://people.csail.mit.edu/mip/papers/structures/paper.pdf, 2010.

APPENDIX

PROOF OF THE DIRECT SUM THEOREM

For completeness, we give a full proof of our direct sum theorem that relates the information complexity of MULTI-AI with that of AI. We begin with a small technical lemma that is an interesting observation in its own right.

Lemma 24. *Let P be a communication protocol involving two players, Alice and Bob, who share a public random string R in addition to their private random strings. Let T denote the transcript of P when Alice receives input X and Bob receives Y, from an arbitrary input distribution. Let A and B denote the portions of T that are communicated by Alice and Bob, respectively. Then*

$$\mathrm{I}(T:X \mid Y,R) = \mathrm{I}(A:X \mid Y,R)\ , \mathrm{I}(T:Y \mid X,R) = \mathrm{I}(B:Y \mid X,R)\,.$$

Proof: By the chain rule for mutual information,

$$\mathrm{I}(T:X \mid Y,R) = \mathrm{I}(A:X \mid Y,R) + \mathrm{I}(B:X \mid A,Y,R)\,.$$

Since Bob's messages are just some function of A,Y,R, and his private coins, for any fixed setting of A,Y,R, then B and X are independent. Thus, $\mathrm{I}(B:X \mid A,Y,R) = 0$. Similarly, we can show that $\mathrm{I}(T:Y \mid X,R) = \mathrm{I}(B:Y \mid X,R)$. ■

Theorem 11 (restated). *Suppose there exists a $[p,s,\varepsilon]$-protocol Q for MULTI-AI$_{m,n}$. Then there exists an ε-error randomized protocol P for AI$_n$ in which Alice sends at most ps bits in total, and which satisfies*

$$m \cdot \mathrm{icost}^B_{\mu_0}(P) \ \leq\ \mathrm{icost}_{\mu_0^{\otimes m}}(Q)\,,$$

where μ_0 is as in Definition 2 and $\mu_0^{\otimes m}$ denotes the m-fold product of μ_0 with itself.

Proof: Using Q, we can derive a family, $\{P_j\}_{j\in[m]}$, of protocols for AI, using the following simulation. Suppose Alice and Bob receive inputs x and $(k,c,x_{1:k-1})$ respectively.

1) Alice sets A_j's input to x and Bob sets B_j's input to $(k,c,x_{1:k-1})$.
2) The players generate $X^1,X^2,\ldots X^{j-1},X^{j+1},\ldots X^m,K^1,\ldots K^{j-1}$ independently and uniformly at random using *public* coins. They choose $C^1,\ldots,C^{j-1}$ so that $X^i_{K^i} = C^i$ for all $i \in [j-1]$. This sets the input to players $A_1,B_1,\ldots,A_{j-1},B_{j-1}$ and ensures that $(X^i,K^i,C^i) \sim \mu_0$ for all $i<j$.
3) Bob generates $K^{j+1},K^{j+2},\ldots,K^m$ independently and uniformly at random using *private* coins. He chooses $C^{j+1},\ldots,C^m$ so that $X^i_{K^i} = C^i$ for each $i \in \{j+1,\ldots,m\}$. This sets the input to players $A_{j+1},B_{j+1},\ldots,A_m,B_m$ and ensures that $(X^i,K^i,C^i) \sim \mu_0$ for all $i>j$.
4) The players now jointly simulate Q on the random input $\mathbf{Z}$ thus generated. In each round:
   a) Alice simulates players $A_1,B_1,\ldots,A_j$ and sends Bob the message that A_j would have sent to B_j.
   b) Bob simulates $B_j,A_{j+1},\ldots,B_m$ and sends Alice the message that B_m would have sent to A_m.
   c) Alice then continues the simulation of $A_m,\ldots,A_1$ and moves on to beginning of the next round (if required), without needing any communication.
5) At the end of the simulation, Alice outputs the answer that player A_1 would have output in Q.

Clearly, Alice communicates at most ps bits in P_j. The definition of μ_0 ensures that AI$(X^i,K^i,C^i) = 0$ for all $i \neq j$, and therefore MULTI-AI$(\mathbf{Z}) =$ AI(X,K,C); thus P_j is correct whenever Q is correct on the randomly generated input. This bounds the worst-case error of P_j by ε. To bound the information cost of P_j, notice that when the input to P_j is distributed according to μ_0, it simulates Q on an input that is distributed according to $\mu_0^{\otimes m}$. Let (X^j,K^j,C^j) denote a random input to P_j distributed according to μ_0, and let T and B denote the resulting random transcript of P_j, and Bob's portion of this transcript, respectively. Defining M_m and R as in (23), we see that $B \equiv M_m$ and that the public random string used by P_j is exactly $R' = (R,\mathbf{X^{-j}},K^1,\ldots,K^{j-1})$. Thus,

$$\begin{aligned}
\mathrm{icost}^B_{\mu_0}(P_j) &= \mathrm{I}(T:K^j,C^j \mid X^j,R') \\
&= \mathrm{I}(B:K^j,C^j \mid X^j,R') \\
&= \mathrm{I}(M_m:K^j,C^j \mid K^1,\ldots,K^{j-1},X^1,\ldots,X^m,R)\,,
\end{aligned}$$

where the second equality follows from Lemma 24. By the chain rule for mutual information, we have

$$\begin{aligned}
&\mathrm{icost}_{\mu_0^{\otimes m}}(Q) \ =\ \mathrm{I}(M_m:K^1,C^1,\ldots,K^m,C^m \mid X^1,\ldots,X^m,R) \\
&= \sum_{j=1}^m \mathrm{I}(M_m:K^j,C^j \mid K^1,C^1,\ldots,K^{j-1},C^{j-1},X^1,\ldots,X^m,R) \\
&= \sum_{j=1}^m \mathrm{I}(M_m:K^j,C^j \mid K^1,\ldots,K^{j-1},X^1,\ldots,X^m,R) \qquad (25) \\
&= \sum_{j=1}^m \mathrm{icost}^B_{\mu_0}(P_j)\,,
\end{aligned}$$

where (25) holds because X^j and K^j completely determine C^j, according to the distribution μ_0. Picking j to minimize $\mathrm{icost}^B_{\mu_0}(P_j)$ now gives us $m \cdot \mathrm{icost}^B_{\mu_0}(P_j) \leq \mathrm{icost}_{\mu_0^{\otimes m}}(Q)$. ■