Trade-off Lower Bounds for Stack Machines

Matei David
Center for Computational Intractability
Princeton University
Princeton, USA

Periklis A. Papakonstantinou
Institute for Theoretical Computer Science
Tsinghua University
Beijing, PR China

***Abstract*—A space bounded Stack Machine is a regular Turing Machine with a read-only input tape, several space bounded read-write work tapes, and an unbounded stack. Stack Machines with a logarithmic space bound have been connected to other classical models of computation, such as polynomial time Turing Machines (P) (Cook; 1971) and polynomial size, polylogarithmic depth, bounded fan-in circuits (NC) e.g., (Borodin et al.; 1989).**

In this paper, we give the first known lower bound for Stack Machines. This comes in the form of a trade-off lower bound between space and number of passes over the input tape. Specifically, we give an explicit permuted inner product function such that any Stack Machine computing this function requires either sublinear polynomial space or sublinear polynomial number of passes. In the case of logarithmic space Stack Machines, this yields an unconditional sublinear polynomial lower bound for the number of passes. To put this result in perspective, we note that Stack Machines with logarithmic space and a single pass over the input can compute Parity, Majority, as well as certain languages outside NC. The latter follows from (Allender; 1989), conditional on the widely believed complexity assumption that EXP is different from PSPACE.

Our technique is a novel communication complexity reduction, thereby extending the already wide range of models of computation for which communication complexity can be used to obtain lower bounds. Informally, we show that a k-player number-in-hand communication protocol for a base function f can efficiently simulate a space- and pass-bounded Stack Machine for a related function F, which consists of several permuted instances of f, bundled together by a combining function h. Trade-off lower bounds for Stack Machines then follow from known communication complexity lower bounds.

The framework for this reduction was given by (Beame and Huynh-Ngoc; 2008), who used it to obtain similar trade-off lower bounds for Turing Machines with a constant number of pass-bounded external tapes. We also prove that the latter cannot efficiently simulate Stack Machines, conditional on the complexity assumption that E is not a subset of PSPACE. It is the treatment of an unbounded stack which constitutes the main technical novelty in our communication complexity reduction.

***Keywords*-AuxPDA; stack; communication complexity; lower bound; space bound; reversals; streaming;**

This work was completed and submitted when both authors were graduate students in the Department of Computer Science, University of Toronto. M.D. is supported in part by the NSF grant CCF-0832797. P.A.P. is supported in part by the National Natural Science Foundation of China Grant 60553001, the National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

I. Introduction

One of the goals of complexity theory is understanding the relative power of various models of computation. Consider the classes P and $\mathsf{LOGSPACE}$ of languages decided by deterministic Turing Machines in polynomial time and logarithmic space, respectively. Also consider $\mathsf{NC} = \bigcup_{i \geq 0} \mathsf{NC}^i$, where NC^i is the class of languages decided by (uniform) circuits of polynomial size, depth $\mathrm{O}\left((\log n)^i\right)$, consisting of bounded fan-in And, Or and Not gates. We regard P and $\mathsf{LOGSPACE}$ as modelling efficient time-bounded and space-bounded computation, respectively, and we regard the class NC as modelling efficient parallelizable computation. We know that

$$\mathsf{NC}^1 \subseteq \mathsf{LOGSPACE} \subseteq \mathsf{NC}^2 \subseteq \ldots \subseteq \mathsf{NC}^i \subseteq \ldots \subseteq \mathsf{P},$$

but we don't know whether any of the inclusions are proper.

One possible way to attack the problem of separating complexity classes defined using different models of computation (e.g., Turing Machines, combinatorial circuits) and different resource bounds (e.g., time, space, size, depth) is to rephrase them as separations of classes based on a common computational model, a common resource bound, and prove lower bounds for that common resource.

Consider the problem of characterizing polynomial-time computation in terms of a space bound. We know that a logarithmic-*space* Turing Machine can be simulated by a polynomial-*time* Turing Machine, and we believe that the opposite is not true in general (i.e., that $\mathsf{LOGSPACE} \subsetneq \mathsf{P}$). Furthermore, we also believe that, e.g., some polylogarithmic-space Turing Machine cannot be simulated by a polynomial-time Turing Machine. Thus, there seems to be no obvious way to exactly capture polynomial-time computation in terms of a space bound.

A *stack*, also called a push-down storage, models an unlimited storage space that comes with a First-In Last-Out access restriction. A *Stack Machine* is a classical Turing Machine equipped with a stack (also called an AuxPDA.) A space bound for a Stack Machine refers exclusively to the size of its work tapes, and not to its stack. In light of the previous paragraph, [1] gives a fascinating characterization of a time-bounded complexity class in terms of a space-bounded computational model, showing that the class of

languages decided by logarithmic-*space* Stack Machines exactly equals that of languages decided by polynomial-*time* Turing Machines, that is, P.

It is not hard to show that there is no loss in computational power in assuming that a logarithmic space Stack Machine also operates in at most exponential ($2^{n^{\mathrm{O}(1)}}$) *time* [5]. A series of subsequent results, e.g. [5], [6], [2], [3], [7], establish a perhaps surprising connection between simultaneously space- and time- bounded Stack Machines and combinatorial circuits: *nondeterministic* logarithmic-space Stack Machines that run in time $2^{\mathrm{O}\left(\log^i n\right)}$ *precisely* characterize SAC^i, the extension of NC^i in which Not gates are at the input level and we allow Or gates with unbounded fan-in. Furthermore, denoting by $\mathsf{SM}(s,t)$ the class of languages decided by deterministic Stack Machines operating in space s and time t, we now know that, for every integer $i \geq 1$,

$$\mathsf{NC}^i \subseteq \mathsf{SM}\left(\mathrm{O}\left(\log n\right), 2^{\mathrm{O}\left((\log n)^i\right)}\right) \subseteq \mathsf{NC}^{i+1} \subseteq \ldots \subseteq \mathsf{P} = \mathsf{SM}\left(\mathrm{O}\left(\log n\right), 2^{n^{\mathrm{O}(1)}}\right).$$

In spite of the connections laid out above between Stack Machines and major open problems in computational complexity, this model is not well understood.

A. Our Contribution

As the main contribution of this work, we provide the first ever lower bounds specifically for Stack Machines. We do not know how to tackle time lower bounds directly. Instead, we consider *the number of (two-way) passes a Stack Machine makes over its input*. We assume that a Stack Machine has at least logarithmic space and, without losing generality, that in every pass, the input head moves from one end of the tape to the other. Sometimes this measure is also called reversal complexity, as the number of passes equals one plus the number of reversals. The lower bounds we prove come in the form of trade-off lower bounds between space and number of passes. Specifically, we give two examples of functions for which any Stack Machine requires either $\Omega\left(N^\beta\right)$ space or $\Omega\left(N^\beta\right)$ passes, for some $\beta < 1$. In the case of logarithmic-space Stack Machines, this translates into unconditional $\Omega\left(N^\beta\right)$ lower bounds for the number of passes.

Communication complexity has been used to derive lower bounds in a wide variety of other areas of theoretical computer science: cell probe complexity, VLSI circuit design, Turing Machine complexity, circuit complexity, pseudorandomness, algorithmic game theory, proof complexity, and streaming. Perhaps as important as its corollaries, our main technical contribution is to show that communication complexity can also be used to derive lower bounds for Stack Machines.

Consider a *base* function $f = f_{k,n} : \left(\{0,1\}^n\right)^k \to \{0,1\}$, and a *combining* function $h : \{0,1\}^m \to \{0,1\}$ that is symmetric and has a neutral element (e.g., this is the case for OR and XOR, with neutral element 0). Let $F = F_{k,m\cdot n} : \left(\left(\{0,1\}^n\right)^m\right)^k \to \{0,1\}$ be a *lifted* function which consists, informally, of m instances of f on disjoint inputs, "permuted" in a way made precise in Section III, and "glued together" by h. Our main technical contribution is the following reduction, saying that a k-player number-in-hand communication protocol for f can efficiently simulate a Stack Machine for F.

Theorem I.1. *Let $k = k(n) \geq 2$ be a non-decreasing function and let $m = m(n) \geq 1$ be an increasing function such that $k \leq m^{\mathrm{O}(1)}$. Let $N = N(n) := k \cdot m \cdot n$. Let $s = s(N)$ and $r = r(N)$ be increasing functions and let $\delta < 1/2$ be a constant. Let $d := k \cdot r \cdot \log r/\sqrt{m}$.*

Let $f = f_{k,n}$ be a boolean base function, and let $F = F_N$ be a function related to f in the sense informally described above, and made precise in Section III.

Assume there exists a randomized (even, nonuniform) Stack Machine for F with space bound s, pass bound r and error δ. Then, there exists a k-player number-in-hand randomized protocol for f, with cost $\mathrm{O}\left(k \cdot r \cdot \log(k \cdot r) \cdot s\right)$ and error at most $\delta + \mathrm{O}\left(d\right)$.

As a consequence of Theorem I.1 and of the known communication complexity lower bound for the Inner Product function, we obtain the following trade-off lower bound, which is the first of its kind.

Corollary I.2 (Informal statement)**.** *Let $\epsilon > 0$ and $\delta < 1/2$ be constants. There exists an "inner product-like" function $F = F_N \in \mathsf{LOGSPACE}$ such that any Stack Machine computing F with error δ requires space $s = \omega\left(N^{1/4-\epsilon}\right)$ or passes $r = \omega\left(N^{1/4-\epsilon}\right)$.*

In particular, a log-space *Stack Machine needs $\omega\left(N^{1/4-\epsilon}\right)$ passes to compute F.*

The ℓ-th frequency moment of a sequence $\bar{a} = (a_1, \ldots, a_t)$, where $a_i \in [R]$, is $\mathrm{Freq}_\ell(\bar{a}) = \sum_{j\in[R]} f_j^\ell$, where $f_j = |\{i \in [t] \mid a_i = j\}|$. Computing Freq_ℓ is a well-studied problem in the streaming literature [8]. As another consequence of Theorem I.1, we also obtain the following result, which can be interpreted as saying that a stack *does not help* in computing frequency moments.

Corollary I.3. *Let $\ell > 4$, $\epsilon \geq 0$ and $\delta < 1/2$ be constants. There exists a constant $0 < \beta < 1$ such that any randomized Stack Machine computing a $(1+\epsilon)$ multiplicative approximation of Freq_ℓ with error δ requires space $s = \omega\left(N'^\beta\right)$ or passes $r = \omega\left(N'^\beta\right)$, where N' denotes the input size.*

In particular, a log-space *Stack Machine needs $\omega\left(N'^\beta\right)$ passes to approximate Freq_ℓ.*

For this result, we use a number-in-hand communication complexity lower bound for the promise Set Intersection function [9], along with a streaming reduction between the problems of computing the promise Set Intersection function

and the frequency moments of a data stream, originating from [8].

Remark I.4. Clearly, the *number of passes* is also a lower bound on the *running time* of a Stack Machine. If interpreted in this way, our method comes with an important limitation. Specifically, by using a reduction to communication complexity it is not clear how to obtain any super-linear lower bounds, because the communication complexity of any function is at most linear. Still, as explained in Section VI, Stack Machines with logarithmic space and few passes over the input are quite powerful, so a lower bound on passes can be seen as interesting in its own right.

Remark I.5. Some of the technical effort in our proofs is directed towards dealing with Stack Machines that have *two-way* rather than *one-way* access to their input tape. We point out that these restrictions are polynomially, but super-linearly, related: a Turing Machine (with or without a stack) that makes $r(N)$ *two-way* passes over an input of size N can be simulated by a similar Turing Machine M' that makes $N \cdot r(N)$ *one-way* passes. However, the method presented in this work can only derive lower bounds of the form $r(N) \geq \Omega\left(N^{\alpha}\right)$ for some $0 < \alpha \leq 1$. Therefore, we cannot use our current methods to first prove lower bounds for one-way access, and then transfer them to two-way access using the simple argument above.

B. Related Work

Turing Machines with limited reversals have been studied before, e.g. [10]. However, in that line of work, reversals are bounded on all tapes. By comparison, the Stack Machines we consider are significantly more powerful, because reversals are unbounded on both their space bounded internal tapes, and their stack.

This type of a reduction, connecting efficient communication protocols and space bounded computation, is not new. One of the first examples is [11], which derives time-space tradeoffs for multi-head Turing Machines. Subsequently, such reductions have been used to derive lower bounds in streaming [8], an area of computer science whose object of study is the power of Turing Machines with small space and a single (or, very few) pass(es) over the input tape.

At the technical level, Stack Machines are related to (r, s, t) read-write stream algorithms. The latter are Turing Machines that have: a constant number t of "external" read-write tapes, several "internal" tapes of combined space s, and a total number r of passes, counted over *all* external tapes. These machines were introduced by [12] as an extension of the standard streaming model, in which the machines has access to a single external read-only tape. [12], [13] derived several lower bounds for deterministic and one-sided error randomized read-write stream algorithms by analyzing their structural properties. [14] showed that two of the standard measures that are used to bound communication complexity, discrepancy and corruption, can be used to derive lower bounds for computing certain direct-sum type functions with two-sided error randomized stream algorithms. Finally, [4] gave a simpler and more direct reduction between number-in-hand communication protocols and read-write stream algorithms, obtaining trade-off lower bounds between space and number of passes, that inspired some the technical arguments in this paper.

Stack Machines and read-write stream algorithms are somewhat similar at the technical level because they both augment a space bounded Turing Machine, with *an unbounded stack* one the one hand, and with *several pass-bounded read-write tapes* on the other. However, the motivation for considering these models is essentially different. Stack Machines are intimately connected to combinatorial circuits, which in turn model efficient parallel computation, whereas read-write stream algorithms model efficient computation in the presence of unlimited but slow "external" memory and fast but limited "internal" memory. In the full version of this paper, we show that read-write stream algorithms cannot efficiently simulate Stack Machines, conditional on the widely believed complexity assumption that $\mathsf{E} \not\subseteq \mathsf{PSPACE}$.

It is the treatment of the stack that constitutes the main technical novelty in our result. To put this into perspective, observe that a Turing Machine equipped with *two* unbounded stacks can decide any decidable language, using no workspace at all and a single pass over the input (e.g., [15, Problem 3.9]). In comparison, Turing Machines with *any constant number* of external tapes with space bound s and pass bound r can only compute languages in $\mathsf{DSPACE}\left(r^2 \cdot s\right)$ [16, Lemma 4.8].

C. Organization

The heart of our result is the a new communication complexity reduction. In Section II, we give an outline of this reduction in a simplified setting which still involves most difficulties. In Section III, we give the formal definitions needed to state our results. In Section IV, we give the formal statements that comprise the reduction. Finally, in Section V, we state the consequences we obtain using the reduction. The formal proofs, including the evidence suggesting that Stack Machines are incomparable to read-write stream algorithms, are deferred to the full version of this paper.

II. Outline of the Argument

Consider the regular Inner Product function $\mathrm{IP} = \mathrm{IP}_{2,n} : \left(\{0,1\}^n\right)^2 \to \{0,1\}$. Let (x_1, x_2) be an input, with $x_p = (x_{p,1}, \ldots, x_{p,n})$, where $x_{p,j} \in \{0,1\}$ for $p \in [2]$ and $j \in [n]$. In the communication complexity world, we are interested in computing this function with 2 player protocols, in which player p gets x_p, for $p \in [2]$. We know that $R_2(\mathrm{IP}_{2,n}) \geq \Omega(n)$ [17]. In the TM world, we are interested in computing

this function when its input is given on a tape in the natural way: first the n bits of x_1, then the n bits of x_2. The input size for the TM is $N = 2 \cdot n$. In this simplified outline, we study the tradeoff between the space $s(N)$ and the number of *one-way passes* $r(N)$ that a *deterministic* TM needs. In the full proof, we allow $k \geq 2$ players, we allow *two-way passes*, and we allow *randomization* in the TM.

In the absence of a stack.: When there is no stack, there is a simple, well-known, simulation of a space and pass bounded TM by a 2 player communication protocol. Player 1 simulates the TM until the head first crosses into the input zone containing x_2, at which point it sends the full state of the TM to player 2. The latter continues the simulation until the head crosses back into x_1, and so on. In total, the communication in this protocol is $\mathrm{O}\left(s(N) \cdot r(N)\right)$. By the communication complexity lower bound, we get $s(N) \cdot r(N) \geq \Omega\left(n\right) = \Omega\left(N\right)$. This simple simulation breaks down in the case of a SM because there might be transfer of information via the stack between the parts of the computation when the input head is scanning x_1 and x_2. One way to see this is to observe that the configuration of a TM can be described by $\mathrm{O}\left(s\right)$ bits, whereas that of a SM might require up to $2^{\Theta(s)}$ bits.

A framework for dealing with a stack.: In order to deal with the presence of a stack, we adapt a framework that was originally introduced by [4] in the context of TMs with several external tapes.

Consider a function f which we know is hard in the communication model. We assume that an efficient SM M exists for a related function F. Using this assumption, we build an efficient communication protocol P for f. Let x be an input to f in P. The players in P first construct an input v to F (which contains x and some public "padding"), they simulate M on v, computing $F(v)$, and finally they derive $f(x)$ from $F(v)$. The simulation of M on v must be efficient so that communication lower bounds apply.

Let Γ be the sequence of configurations of M on v. In P, the players simulate Γ using a series of publicly and privately simulated sections. The public simulation is "cheap" in the sense that it does not cost any communication. The private simulation is "expensive" because, at the end of each private section, the player simulating it must communicate something in order for the other player(s) to "know where they are" in Γ. Each section in Γ where the input head scans a symbol from x (which is the input to P) is to be simulated privately by the player who knows that symbol. Moreover, the basic idea is that, since we do not have any bounds on how a SM can use its stack, we want to avoid communicating stack content in P. Thus, we do not mind if either: a symbol is pushed on the stack during public simulation and popped during either public or private simulation; or a symbol is pushed on the stack in a privately simulated section by player p and later popped in a (possibly different) privately simulated section the same player p. What we want to avoid is the remaining scenario: a symbol is pushed on the stack in a privately simulated section by player p_1 and later popped in a privately simulated section by player $p_2 \neq p_1$. Informally, the way we achieve this "protection" against M using its stack on x is by "hiding" x into the larger input v, so that, with high probability, M only uses its stack for "meaningless" computation.

Concretely, let $m = n = \sqrt{N/2}$. Suppose that in a communication protocol 2 players want to compute $\mathrm{IP}_{2,n}$, and they have access to an efficient (space $s(N)$, one-way passes $r(N)$) SM for $\mathrm{IP}_{2,N/2}$. We think of $\mathrm{IP}_{2,N/2}$ as m instances of $\mathrm{IP}_{2,n}$ glued together by a top-level XOR_m gate.

> In the communication protocol P, the players get inputs $x_1, x_2 \in \{0,1\}^n$, respectively. They begin by choosing a random $i^* \in [m]$ and random $y_{1,i}, y_{2,i} \in \{0,1\}^n$ for $i \in [m] \setminus \{i^*\}$, all using public coins. Let $v = (v_1, v_2)$ be the input to M defined by, $v_{p,i^*} := x_p$ and $v_{p,i} := y_{p,i}$, for $i \neq i^*$ and $p \in [2]$. Henceforth, the goal of P is to simulate M on v.

Thus, both players know most of the input to the SM, except for two pieces of n bits each, where their respective inputs from the communication protocol are embedded in the N bit input v to M. Furthermore, observe that if they were indeed able to simulate the SM, they could compute the output for the protocol as $\mathrm{IP}(x_1, x_2) = \mathrm{IP}(v_{1,i^*}, v_{2,i^*}) = \mathrm{XOR}_2\left(\mathrm{IP}(v_1, v_2), \mathrm{XOR}_{m-1}\left(\mathrm{IP}(v_{1,i}, v_{2,i}) \mid i \neq i^*\right)\right)$.

Unique stack symbols.: For the purposes of the analysis, we assume that each symbol on the stack is given a unique tag. Thus, even though the same symbol might appear many times on the stack, we assume we can distinguish between any of those appearances. In particular, to say that "a symbol is placed on the stack in configuration γ_1 and popped in configuration γ_2" formally means that: the stack level is the same in γ_1 and in the configuration immediately following γ_2; and the stack level is strictly higher in any intermediate configuration between those two.

Corrupted instances.: A key concept in our argument is that of a "corrupted instance". For a SM M and an input v, we say that *instance i is corrupted in v on M* if, during the run of M on v, a symbol is placed on the stack while the input head is scanning $v_{p_1,i}$ and popped while it is scanning $v_{p_2,i}$, where $p_1 \neq p_2$.

Three claims.: Henceforth, our argument is built on the following three claims:

(i) The number of corrupted instances in any one input is *small*.
(ii) The choice of i^* and of v in the communication protocol are *statistically independent* of each other.
(iii) If the input x to the communication protocol *is not* embedded in a corrupted instance in the input v to the SM, then the protocol can *efficiently* simulate the SM.

The intuition for claim (i).: Fix a SM M and an input v. We want to give a bound for the number of corrupted instances in v on M. We associate each corrupted instance $i \in [m]$ with a unique 4-tuple $(l_1 \leq l_2, p_1 \neq p_2)$, such that a symbol is pushed on the stack in the pass l_1 over $v_{p_1,i}$ and popped in the pass l_2 over $v_{p_2,i}$. Note, for each corrupted instance there might be several such 4-tuples to choose from, here, we associate i with only (any) one of them. We say that i *is corrupted* by this 4-tuple.

Let $l_1 \leq l_2 \leq r$ and let $p_1 \neq p_2 \in [2]$. During the pass l_1 over v_{p_1}, the input head scans the instances in v going from left to right in the order: $1, 2, \ldots, m$. The same is true for the pass l_2 over v_{p_2}. Assume instances $i_1 \neq i_2$ are corrupted by this particular 4-tuple. Without losing generality, say $1 \leq i_1 < i_2 \leq m$. We see that: the stack symbol associated with i_1 is pushed first, then the stack symbol associated with i_2, then the stack symbol associated with i_1 is popped, and finally the stack symbol associated with i_2 is popped. This contradicts the First-In Last-Out access semantics of a stack. Hence, at most one instance is corrupted by any one 4-tuple.

Since the number of 4-tuples is at most $2 \cdot r^2$, we derive that at most $\mathrm{O}\left(r^2\right)$ instances can be corrupted in v on M. Thus, as long as $r^2/m = o\left(1\right)$, the fraction of corrupted instances in any one input is small. This argument is extended to work for *two-way* access, and slightly improved, as Lemma IV.1.

The intuition for claim (ii).: To see why we need (ii), observe that in the communication protocol P, v is constructed *based on* i^*. For (i) to be useful, we would like the choice of i^* to probabilistically "hide" the instance containing the input to P from the set of corrupted instances in v on M. It seems that we would need i^* to be chosen *after* v.

To achieve (ii), we use an argument that goes through distributional communication complexity: for every distribution D on the inputs to $\mathrm{IP}_{2,n}$ in the communication protocol, the players choose the $m-1$ "decoy" instances from that same distribution D, so that v is distributed according to D^m *independently of* i^*. In this case, we can permute the choices of v and i^*, so that claim (i) gives a bound on the probability the communication protocol embeds the real input in a corrupted instance. We go back to randomized communication complexity (for which we have lower bounds) using the standard Yao's min-max principle [18, Theorem 3.20] connecting these two measures. This argument is made precise inside the proof of Theorem I.1.

The intuition for claim (iii).: Assume that the instance i^*, where the real input $x = (x_1, x_2)$ to the communication protocol P is embedded in v, is *not* corrupted. We argue that P can efficiently simulate M.

Let Γ be the sequence of configurations that M goes through on input v. We say that a configuration is *input-private to player* p if the input head is scanning a symbol from v_{p,i^*} (which is where x_p is embedded). Intuitively, we want player p to simulate the transition out of a configuration that is input-private to player p, because only it knows the symbols in $v_{p,i^*} = x_p$. Moreover, symbols might be pushed on the stack in such a transition, we say that any such stack symbol is *private to player* p. We say that a configuration is *stack-private to player* p if the transition out of this configuration pops a stack symbol that is private to player p. Intuitively, we want player p to simulate such a transition in order to avoid communicating stack contents during the protocol. All other configurations are *public*. Since we assumed instance i^* is not corrupted, we know there is no configuration which is input- and stack-private to different players.

The players simulate Γ by alternating between public and private simulation. At the end of each privately simulated section, the player performing that simulation communicates: the state of the SM, the lowest stack level reached during the private simulation section, the current stack level, and the top stack symbol. It can be shown that *the cost of the communication at the end of each privately simulated section is* $\mathrm{O}\left(s\right)$. In the proof of Lemma IV.2, we argue that the information communicated is sufficient for the players to obtain a "hollow view" of the stack: each player knows the public stack symbols, the stack symbols which are private to itself, and the locations of the stack symbols private to other players. Moreover, this hollow view is sufficient for the players to continue the simulation until the end of Γ.

Finally, let us give an informal bound on the amount of communication. Observe that input-private configurations form exactly $2 \cdot r$ contiguous subsequences in Γ. Let us denote these contiguous zones of input-private configurations in Γ by S_a, for $1 \leq a \leq 2 \cdot r$. Let γ_a and γ'_a be the configurations at the beginning and end of S_a, respectively. Observe that the stack at the end of S_a (in γ'_a) contains at most a *contiguous zones of private stack symbols*, at most one such zone corresponding to each previous input-private section $S_{a'}$, with $a' \leq a$. Some of these zones might be accessed before S_{a+1}. As explained in detail in the proof of Lemma IV.2, the players in the protocol can simulate the sequence of configurations between γ'_a and γ_{a+1} using *as many privately simulated sections as there are contiguous zones of private stack symbols in* γ'_a. Since $a \leq 2 \cdot r$, the number of privately simulated sections between γ'_a and γ_{a+1} is $\mathrm{O}\left(r\right)$. Summing over all a, and taking into account the privately simulated sections S_a, we get that, in total, the simulation of Γ can be performed using at most $\mathrm{O}\left(r^2\right)$ privately simulated sections. Hence, the communication bound for the entire protocol is $\mathrm{O}\left(r^2 \cdot s\right)$.

Putting (i), (ii) and (iii) together, assuming that we have a SM for $\mathrm{IP}_{2,N/2}$, we obtain a randomized communication protocol for $\mathrm{IP}_{2,n}$ that has error $\mathrm{O}\left(r^2/m\right)$ and cost $\mathrm{O}\left(r^2 \cdot s\right)$. As long as $r^2/m = o\left(1\right)$, the known randomized communication complexity lower bound $R_2(\mathrm{IP}_{2,n}) \geq \Omega\left(n\right)$ applies [17], and we obtain the tradeoff lower bound $r(N)^2 \cdot$

$s(N) \geq \Omega(n) = \Omega\left(N^{1/2}\right)$.

What was left out of this outline.: First, *the number of players* in this outline is set to $k = 2$. While this is sufficient to derive SM tradeoff lower bounds for Inner Product-like functions, we need to allow the number of players to be a function of the input size, that is, $k = k(n)$, in order to obtain inapproximability results for the frequency moments problem in Corollary I.3.

Second, the Inner Product function is of such a nature that no matter how the players in P choose the decoy instances $y_{p,i} \in \{0,1\}^n$ for $p \in [2]$ and $i \neq i^*$, they can always retrieve $f(x) = \mathrm{IP}(x_1, x_2)$ from $F(v) = \mathrm{IP}(v_1, v_2)$ by computing the XOR of all decoy instances $\mathrm{IP}(y_{1,i}, y_{2,i})$ together with $F(v)$. This is a property of the top-level gate in IP, which is XOR. In order to deal with the frequency moments problem, we use a reduction involving the promise Set Intersection problem, which has top-level gate OR. In this case, in order to be able to retrieve $f(x)$ from $F(v)$, the decoy instances *must* be 0-instances, and only then do we have $f(x) = F(v)$. To accommodate for this requirement and still achieve claim (ii), we need to treat 0-inputs and 1-inputs differently inside the proof of Theorem I.1.

Third, we need to fill in the details of the simulation informally described as claim (iii).

Fourth, there is the issue of *two-way versus one-way passes*. This outline only considers how to obtain trade-off lower bounds for SMs with one-way access to the input. As we have seen above, a SM with two-way access to the input can compute IP with only 2 passes. The point where this framework breaks down is claim (i), which is outright false. Consider the case when $p_1 = 1$, $p_2 = 2$, the pass $l_1 = 1$ is left-to-right, and the pass $l_2 = 2$ is right-to-left. We see that the instances are visited in the pass 1 in the order $1, 2, \ldots, m-1, m$ and they are visited in the pass 2 in the order $m, m-1, \ldots, 2, 1$. Then, the associated "corrupting" stack symbols can be pushed in the order $1, \ldots, m$ and popped in the order $m, \ldots, 1$, without violating the access semantics of a stack. Thus, potentially *all* instances can be corrupted by a single 4-tuple $(1, 2, 1, 2)$, which is precisely what happens in the simple 2-pass protocol seen earlier.

In order to obtain tradeoff lower bounds for SMs with two-way access, we need to fix claim (i). To that end, we "scramble" the order in which the m instances are seen in pass l_1 over v_{p_1} and in pass l_2 over v_{p_2}, no matter what p_1 and p_2 are, and no matter whether the passes l_1 and l_2 are left-to-right or right-to-left. Informally, we achieve this by reordering the m instances corresponding to each player using a family of permutations $\Phi = (\varphi_1, \ldots, \varphi_k)$, with one permutation for each player, that has the following property.

> For $p \in [k]$, let $\sigma_p := (\varphi_p(1), \ldots, \varphi_p(m))$ and $\sigma_p^{\mathrm{rev}} := (\varphi_p(m), \ldots, \varphi_p(1))$. Then, for every $p_1 \neq p_2$, $\sigma_1' \in \{\sigma_{p_1}, \sigma_{p_1}^{\mathrm{rev}}\}$, and $\sigma_2' \in \{\sigma_{p_2}, \sigma_{p_2}^{\mathrm{rev}}\}$, σ_1' and σ_2' have *as small a common subsequence as possible*.

It turns out that there exist families Φ where any such common subsequence has size $\mathrm{O}(\sqrt{m})$ [4].

III. Definitions and Facts

We use standard definitions for communication complexity and for Turing Machines.

We use n to denote the size of the input to a player in a communication protocol. The number of players is $k = k(n)$, which is a non-decreasing function of n. We use $f = f_{k,n} : (\{0,1\}^n)^k \to \{0,1\}$ to denote a generic boolean function which we are interesting in computing in the communication model. We use $F : \{0,1\}^N \to \{0,1\}$ to denote a generic function that we are interested in computing in the Stack Machine model. We denote by N the size of the input to F. We use $s = s(N)$ and $r = r(N)$ to denote increasing functions bounding the space and the number of passes of a Stack Machine.

The Inner Product function $\mathrm{IP}_{2,n} : (\{0,1\}^n)^2 \to \{0,1\}$ is defined by $\mathrm{IP}_{2,n}(x_1, x_2) = \sum_{i=1}^n x_{1,i} \cdot x_{2,i}$. The Set Intersection function $\mathrm{SetInt}_{k,n} : (\{0,1\}^n)^k \to \{0,1\}$ is defined by $\mathrm{SetInt}_{k,n}(x_1, \ldots, x_k) = \bigvee_{i=1}^n \left(\wedge_{j=1}^k x_{j,i}\right)$. In the *promise* Set Intersection function $\mathrm{pSetInt}_{k,n}$, we are promised that the inputs satisfy the following: there is at most one $i \in [n]$ such that $\wedge_{j=1}^k x_{j,i} = 1$; and for every $i \in [n]$ where $\wedge_{j=1}^k x_{j,i} = 0$, there is at most one $j \in [k]$ such that $x_{j,i} = 1$.

Permutations, Sequences, and Sortedness.: Let S_m denote the set of all permutations of $[m]$. Let id denote the identity permutation. For a permutation $\pi \in S_m$, let $\mathrm{seq}(\pi)$ denote the m-element sequence $(\pi(1), \pi(2), \ldots, \pi(m))$. For a sequence σ, let σ^{rev} denote σ reversed. E.g., $\mathrm{seq}(\pi)^{\mathrm{rev}} = (\pi(m), \ldots, \pi(1))$. For two sequences σ_1, σ_2, let $\mathrm{LCS}(\sigma_1, \sigma_2)$ denote their *longest common* (not necessarily contiguous) *subsequence*. For a sequence σ, let $|\sigma|$ denote its length.

Consider two permutations $\varphi_1, \varphi_2 \in S_m$. We define their *relative sortedness* as

$$\mathrm{relsort}(\varphi_1, \varphi_2) := \max |\mathrm{LCS}(\sigma_1, \sigma_2)|, \quad \forall i \in [2], \forall \sigma_i \in \{\mathrm{seq}(\varphi_i), \mathrm{seq}(\varphi_i)^{\mathrm{rev}}\}$$

For a set of permutations $\Phi = \{\varphi_1, \ldots, \varphi_t\}$, we define its relative sortedness to be the maximum relative sortedness of any two of its members. Formally, $\mathrm{relsort}(\Phi) := \max\{\mathrm{relsort}(\varphi_i, \varphi_j) \mid i \neq j \in [t]\}$.

The following facts give us sets of permutations with small relative sortedness.

Fact III.1 (Lemma 6 in [12])**.** *For every m, there exists a permutation $\psi_m^* \in S_m$ such that* $\mathrm{relsort}(\mathrm{id}, \psi_m^*) \leq 2 \cdot \sqrt{m} - 1$. *Furthermore, ψ_m^* can be computed in space* $\mathrm{O}(\log m)$.

[4] use a simple counting argument to show that a similar bound, still asymptotically optimal, can be achieved even for sets of permutations.

Fact III.2 (Corollary 2.2 in [4]). *Let $k = k(m)$ be a function such that $k \leq m^{\mathrm{O}(1)}$. There exists a family $\Phi^* = \left(\Phi^*_{k,m}\right)_m$ where $\Phi^*_{k,m} = \{\varphi_1, \ldots, \varphi_k\}$ is a set of k permutations from S_m, such that* $\mathrm{relsort}\left(\Phi^*_{k,m}\right) \leq \mathrm{O}\left(\sqrt{m}\right)$.

Permuted Functions.: Let $f = f_{k,n} : \left(\{0,1\}^n\right)^k \to \{0,1\}$ be a *base function*. Let $h = h_m : \{0,1\}^m \to \{0,1\}$ be a *combining function*. Let $\Phi = \Phi_{k,m} = \{\varphi_1, \ldots, \varphi_k\}$ be a *set of k permutations* from S_m. In what follows, we define $\mathrm{LiftMix}\left(f_{k,n}, h_m, \Phi_{k,m}\right) : \left(\{0,1\}^{m \cdot n}\right)^k \to \{0,1\}$ which is a *lift-and-mix* function that consists of m *instances of f, permuted by Φ, and combined by h*.

Let $v \in \left(\{0,1\}^{m \cdot n}\right)^k$ be an input. Let $v = (v_1, \ldots, v_k)$, where $v_p \in \{0,1\}^{m \cdot n}$ for every $p \in [k]$. Let $v_p = (v_{p,1}, \ldots, v_{p,m})$, where $v_{p,i} \in \{0,1\}^n$ for every $(p,i) \in [k] \times [m]$. Let $v_{p,i} = (v_{p,i,1}, \ldots, v_{p,i,n})$, where $v_{p,i,j} \in \{0,1\}$ for every $(p,i,j) \in [k] \times [m] \times [n]$. When v is given as an input to a TM, v appears on the input tape as $v_{1,1}, \ldots, v_{1,m}, v_{2,1}, \ldots, v_{2,m}, \ldots, v_{k,1}, \ldots, v_{k,m}$, and for $(p,i) \in [k] \times [m]$, $v_{p,i}$ appears as the sequence of bits $v_{p,i,1}, v_{p,i,2}, \ldots, v_{p,i,n}$.

For $(p,i') \in [k] \times [m]$, we say that $v_{p,\varphi_p^{-1}(i')}$ is *the i'-th instance* (of f) *inside* v_p. For $i' \in [m]$, we define *the i'-th instance* (of f) *inside* v to be

$$v_{[i'],\Phi} := \left(v_{1,\varphi_1^{-1}(i')}, v_{2,\varphi_2^{-1}(i')}, \ldots, v_{k,\varphi_k^{-1}(i')}\right) \in \left(\{0,1\}^n\right)^k.$$

For $(p,i) \in [k] \times [m]$, $v_{p,i}$ appears in $v_{[i'],\Phi}$ if and only if $i = \varphi_p^{-1}(i')$. Since this is equivalent to $i' = \varphi_p(i)$, we observe the following basic fact.

Fact III.3. *The order on the tape of the m instances inside v_p is $(\varphi_p(1), \varphi_p(2), \ldots, \varphi_p(m))$, which is precisely* $\mathrm{seq}(\varphi_p)$. *Thus, a left-to-right tape scan over v_p visits the m instances in the order* $\mathrm{seq}(\varphi_p)$, *and a right-to-left tape scan visits them in the order* $\mathrm{seq}(\varphi_p)^{\mathrm{rev}}$.

We now define the lift-and-mix function $\mathrm{LiftMix}\left(f, h, \Phi\right) : \left(\{0,1\}^{m \cdot n}\right)^k \to \{0,1\}$ by

$$\mathrm{LiftMix}\left(f, h, \Phi\right)(v) := h(f(v_{[1],\Phi}), f(v_{[2],\Phi}), \ldots, f(v_{[m],\Phi})).$$

We embed an input to f into a specific instance of an input to F as follows. Let $x \in \left(\{0,1\}^n\right)^k$ be an input to f. Let $i^* \in [m]$ be an instance number. Let $\bar{y}_{-i^*} = (y_1, \ldots, y_{i^*-1}, y_{i^*+1}, \ldots, y_m)$ be a set of $m-1$ inputs to f, where $y_i \in \left(\{0,1\}^n\right)^k$ for $i \neq i^*$. We define $v_\Phi(i^*, x, \bar{y}_{-i^*})$ to be the input to F in which x is embedded at instance i^* and y_i is embedded at instance i, for $i \neq i^*$. Formally,

$$v_\Phi(i^*, x, \bar{y}_{-i^*}) := v \in \left(\{0,1\}^{m \cdot n}\right)^k, \text{ such that } \begin{cases} v_{[i],\Phi} = x, & i = i^* \\ v_{[i],\Phi} = y_i, & i \neq i^* \end{cases}$$

Corrupted Instances.: Let M be a deterministic SM and v be an input of size $N = k \cdot m \cdot n$. We say that *instance $i \in [m]$ is corrupted in v on M* if there exist players $p_1 \neq p_2 \in [k]$ such that, during the run of M on v, a symbol is pushed on the stack when the input head is scanning $v_{p_1,\varphi_{p_1}^{-1}(i)}$ and that symbol is popped when the input head is scanning $v_{p_2,\varphi_{p_2}^{-1}(i)}$, where $\varphi_{p_1}, \varphi_{p_2} \in \Phi_{k,m}$. Observe that both strings above are part of $v_{[i],\Phi}$, which is instance i inside v.

Let $\mathrm{BAD}(M, v) \subseteq [m]$ be the set of all instances which are corrupted in v on M. Observe that this definition implicitly depends on the values of k, m, n and Φ, so we should formally write $\mathrm{BAD}_{k,m,n,\Phi}(M, v)$. We use the shorter form for brevity.

Neutral Element.: We say that a combining function $h : \{0,1\}^m \to \{0,1\}$ *has a neutral element* $e \in \{0,1\}$ if (i) $h(e^m) = 0$ and (ii) if an input w has exactly $m-1$ elements set to e then $h(w)$ equals the number of bits that are different from e in w (this can only be 0 or 1). Observe that both OR and XOR have neutral element $e = 0$.

IV. The Reduction

Theorem I.1 (Restated). *Let $k = k(n) \geq 2$ be a non-decreasing function and let $m = m(n) \geq 1$ be an increasing function such that $k \leq m^{\mathrm{O}(1)}$. Let $N = N(n) := k \cdot m \cdot n$. Let $s = s(N)$ and $r = r(N)$ be increasing functions. Let $\delta < 1/2$ be a constant. Let $\Phi = \left(\Phi_{k,m}\right)_m$ be a family of permutations, where $\Phi_{k,m} = \{\varphi_1, \ldots, \varphi_k\}$ are k permutations from S_m. Let $d := k \cdot r \cdot \log r \cdot \mathrm{relsort}\left(\Phi_{k,m}\right)/m$.*

Let $f = f_{k,n}$ be a boolean base function and let $h = h_m$ be a combining function with a neutral element. Let $F = F_N := \mathrm{LiftMix}\left(f_{k,n}, h_m, \Phi_{k,m}\right)$.

Assume there exists a randomized nonuniform SM M for F with space bound s, pass bound r and error δ. Then, there exists a randomized k-player NIH communication protocol P for f, with cost $\mathrm{O}\left(k \cdot r \cdot \log(k \cdot r) \cdot s\right)$ *and error at most* $\delta + \mathrm{O}\left(d\right)$.

To prove this Theorem, use the following two Lemmas, which correspond to claims (i) and (iii) in the outline from Section II. Throughout this section, let n, k, m, N, s, r, Φ be as in Theorem I.1.

The first Lemma gives a bound on the number of corrupted instances inside a fixed input. As such, it directly corresponds to claim (i) from the outline in Section II. As opposed to that claim, the Lemma allows for a growing number of players $k = k(n)$ and for two-way passes over the input tape thanks to the "scrambling" of instances according to the family of permutations Φ.

Lemma IV.1. *Let M' be a deterministic nonuniform SM with space bound s and (two-way) pass bound r. Let v' be an input of size N. Then, $|\mathrm{BAD}(M', v')| \leq \mathrm{O}\left(k \cdot r \cdot \log r \cdot \mathrm{relsort}\left(\Phi_{k,m}\right)\right)$.*

The next Lemma corresponds to claim (iii) from the outline in Section II. It says that if the players in a communication protocol embed their input in a non-corrupted instance, then the protocol can efficiently simulate the SM.

Lemma IV.2. *Let M' be a deterministic nonuniform SM. Let $i^* \in [m]$ be an instance number and let $\bar{y}_{-i^*} = (y_1, \ldots, y_{i^*-1}, y_{i^*+1}, \ldots, y_m)$ be a set of $m-1$ inputs to f. Then, there exists a deterministic k-player NIH communication protocol $P' = P'(M', i^*, \bar{y}_{-i^*})$ such that, on input x:*

- *if $i^* \in \mathrm{BAD}(M', v(i^*, x, \bar{y}_{-i^*}))$, then P' outputs "fail";*
- *otherwise, P' correctly simulates M' and outputs $M'(v(i^*, x, \bar{y}_{-i^*}))$;*
- *the cost of P' is $\mathrm{O}\left(k \cdot r \cdot \log(k \cdot r) \cdot s\right)$.*

V. The Consequences

Corollary I.2 (Restated)**.** *Let $\epsilon > 0$ and $\delta < 1/2$ be constants. There exists a constant $\alpha = \alpha(\epsilon) > 0$ such that the following holds. Let $m = m(n) := n^{\alpha}$, let $k = k(n) := 2$, and let $N := 2 \cdot m \cdot n$. Let $\Psi^*_{2,m} := \{\mathrm{id}, \psi^*_m\}$ for the permutation ψ^*_m defined in Fact III.1. Let $F = F_N := \mathrm{LiftMix}\left(\mathrm{IP}_{2,n}, \mathrm{XOR}_m, \Psi^*_{2,m}\right)$.*

Every randomized SM that computes F with error δ requires space $s = \omega\left(N^{1/4-\epsilon}\right)$ or two-way passes $r = \omega\left(N^{1/4-\epsilon}\right)$.

Observe that, by Fact III.1, the function $F \in \mathsf{LOGSPACE}$.

The proof of Corollary I.2 mainly consists of setting the right parameters in order to apply Theorem I.1. The conclusion follows from the known communication complexity lower bound $R_2(\mathrm{IP}_{2,n}) \geq \Omega\left(n\right)$ [17].

Corollary I.3 (Restated)**.** *Let $\ell > 4$, $\epsilon \geq 0$ and $\delta < 1/2$ be constants. There exists a constant $0 < \beta < 1$ such that any randomized SM computing a $(1+\epsilon)$ multiplicative approximation of Freq_ℓ with error δ requires space $s = \omega\left(N'^{\beta}\right)$ or passes $r = \omega\left(N'^{\beta}\right)$, where N' denotes the input size.*

In order to prove Corollary I.3, we use one more result, saying that an efficient SM A computing frequency moments can be transformed into an efficient SM B computing the permuted Set Intersection function $F_N = \mathrm{LiftMix}\left(\mathrm{pSetInt}_{k,n}, \mathrm{OR}_m, \Phi_{k,m}\right)$, for any permutation family $\Phi = (\Phi_{k,m})_m$. Subsequently, we apply Theorem I.1 to obtain an efficient NIH protocol for the promise Set Intersection function for which we have the lower bound $R_k(\mathrm{pSetInt}_{k,n}) \geq \Omega\left(n/k\right)$ [9].

This following streaming reduction originates in [8], where there are no permutations Φ to deal with. It was rewritten in [4] to deal with the case of several external tapes, where the permutations Φ are needed. The version in here is a combination of the two: we need to deal with permutations, and we also have to perform it "online", because we have no external tapes in this model. In order to deal with the permutations Φ, we use non-uniformity.

Lemma V.1. *Let $\ell > 1$, $\epsilon \geq 0$ and $0 < \delta < 1/2$ be constants. Let n, k, m, N, s, r, Φ be as in Theorem I.1. Set $k = k(n) := \Theta\left((m \cdot n)^{1/\ell}\right)$. Let $F_N := \mathrm{LiftMix}\left(\mathrm{pSetInt}_{k,n}, \mathrm{OR}_m, \Phi_{k,m}\right)$.*

Assume there exists a randomized SM A with space bound s, pass bound r, and error δ, that computes a $(1+\epsilon)$ multiplicative approximation of Freq_ℓ. Then, there exists a randomized nonuniform SM $B = (B_N)$ that computes F_N with space bound $s'(N) = s(N) + \mathrm{O}\left(\log N\right)$, pass bound $r'(N) = r(N) + 2$, and error δ.

VI. Discussion

When considering logarithmic space Stack Machines, the lower bounds that we prove on the number passes over the input tape are only sublinear, and they become uninteresting if directly translated into time lower bounds. However, we believe it is interesting to interpret these results *without* attempting to translate them into time lower bounds.

To see this, consider the class $\mathcal{C}$ of languages decidable by logarithmic space Stack Machines with *a single pass over the input*. On the one hand, Corollaries I.2 and I.3 give examples of functions unconditionally outside $\mathcal{C}$. On the other, it is not hard to see the following fact (we defer its formal proof to the full version).

Fact VI.1. *$\mathcal{C}$ contains all unary languages (tally sets) in P.*

This fact highlights the power of the class $\mathcal{C}$. By [3], $\mathcal{C}$ contains languages outside NC, conditional on $\mathsf{EXP} \neq \mathsf{PSPACE}$. In the full version of this paper, we show that $\mathcal{C}$ contains languages outside $\mathsf{POLYLOGSPACE}$, conditional on $\mathsf{E} \not\subseteq \mathsf{PSPACE}$.

Logarithmic space Stack Machines are closely connected to combinatorial circuits. As such, we ask whether such machines restricted to a sublinear polynomial number of passes characterize (even partially) a natural circuit family. We point out that this family has the following properties: (i) the circuits are of polynomial size; (ii) the family can compute Parity and Majority; and (iii) the family can compute some languages outside NC (again, assuming $\mathsf{EXP} \neq \mathsf{PSPACE}$).

Acknowledgements.

We would like to thank Siavosh Benabbas, Toni Pitassi and Charlie Rackoff for helpful discussions.

References

[1] S. A. Cook, "Characterizations of pushdown machines in terms of time-bounded computers," *J. Assoc. Comput. Mach.*, vol. 18, pp. 4–18, 1971.

[2] A. Borodin, S. A. Cook, P. Dymond, L. Ruzzo, and M. Tompa, "Two applications of inductive counting for complementation problems," *SIAM J. Comput.*, vol. 18, 1989.

[3] E. W. Allender, "P-uniform circuit complexity," *J. Assoc. Comput. Mach.*, vol. 36, no. 4, pp. 912–928, 1989.

[4] P. Beame and D.-T. Huynh-Ngoc, "On the value of multiple read/write streams for approximating frequency moments," in *FOCS*. IEEE, 2008, pp. 499–508.

[5] W. L. Ruzzo, "Tree-size bounded alternation," *J. Comput. Syst. Sci.*, vol. 21, no. 2, pp. 218–235, 1980.

[6] ——, "On uniform circuit complexity," *J. Comput. Syst. Sci.*, vol. 22, no. 3, pp. 365–383, 1981.

[7] H. Venkateswaran, "Properties that characterize LOGCFL," *J. Comput. System Sci.*, vol. 43, no. 2, pp. 380–404, 1991.

[8] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 137–147, 1999.

[9] A. Gronemeier, "Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness," in *STACS*, 2009, pp. 505–516.

[10] J.-e. Chen and C.-K. Yap, "Reversal complexity," *SIAM J. Comput.*, vol. 20, no. 4, pp. 622–638, 1991.

[11] L. Babai, N. Nisan, and M. Szegedy, "Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs," *J. Comput. Syst. Sci.*, vol. 45, no. 2, pp. 204–232, 1992.

[12] M. Grohe and N. Schweikardt, "Lower bounds for sorting with few random accesses to external memory," in *PODS*. ACM, 2005, pp. 238–249.

[13] M. Grohe, A. Hernich, and N. Schweikardt, "Randomized computations on large data sets: Tight lower bounds," in *PODS*. ACM, 2006, pp. 243–252.

[14] P. Beame, T. S. Jayram, and A. Rudra, "Lower bounds for randomized read/write stream algorithms," in *STOC*. ACM, 2007, pp. 689–698.

[15] M. Sipser, *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[16] A. Hernich and N. Schweikardt, "Reversal complexity revisited," *Theor. Comput. Sci.*, vol. 401, no. 1-3, pp. 191–205, 2008.

[17] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM J. Comput.*, vol. 17, no. 2, pp. 230–261, Apr. 1988.

[18] E. Kushilevitz and N. Nisan, *Communication complexity*. New York, NY, USA: Cambridge University Press, 1997.