Small Circuits Imply Efficient Arthur-Merlin Protocols

Michael Ezra
Department of Computer Science, Technion, Haifa, Israel

Ron D. Rothblum
Department of Computer Science, Technion, Haifa, Israel

Abstract

The inner product function $\langle x, y\rangle = \sum_i x_i y_i \bmod 2$ can be easily computed by a (linear-size) $\mathsf{AC}^0(\oplus)$ circuit: that is, a constant depth circuit with AND, OR and parity (XOR) gates. But what if we impose the restriction that the parity gates can only be on the bottom most layer (closest to the input)? Namely, can the inner product function be computed by an AC^0 circuit composed with a single layer of parity gates? This seemingly simple question is an important open question at the frontier of circuit lower bound research.

In this work, we focus on a minimalistic version of the above question. Namely, whether the inner product function cannot be *approximated* by a small DNF augmented with a single layer of parity gates. Our main result shows that the existence of such a circuit would have unexpected implications for interactive proofs, or more specifically, for interactive variants of the Data Streaming and Communication Complexity models. In particular, we show that the existence of such a small (i.e., polynomial-size) circuit yields:

1. An $O(d)$-message protocol in the Arthur-Merlin Data Streaming model for every n-variate, degree d polynomial (over $\mathbb{GF}(2)$), using only $\widetilde{O}(d) \cdot \log(n)$ communication and space complexity. In particular, this gives an $\mathsf{AM}[2]$ Data Streaming protocol for a variant of the well-studied triangle counting problem, with poly-logarithmic communication and space complexities.
2. A 2-message communication complexity protocol for any sparse (or low degree) polynomial, and for any function computable by an $\mathsf{AC}^0(\oplus)$ circuit. Specifically, for the latter, we obtain a protocol with communication complexity that is poly-logarithmic in the size of the $\mathsf{AC}^0(\oplus)$ circuit.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Circuits Complexity, Circuit Lower Bounds, Communication Complexity, Data Streaming, Arthur-Merlin games, Interactive Proofs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.67

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2021/127/download` [47]

Acknowledgements We thank Yuval Ishai, Eyal Kushilevitz and Or Meir for very useful discussions and comments.

1 Introduction

Understanding the expressive power of bounded depth circuits is a central goal in complexity theory, with the hope of eventually answering fundamental questions, such as $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{P} \not\subseteq \mathsf{NC}_1$. Seminal works from the 80's showed that the parity function cannot be computed by AC^0 circuits - that is, constant-depth polynomial-size circuits with unbounded fan-in AND, OR and NOT gates [23, 2, 29]. Razborov and Smolensky [44, 51] took the next step forward by considering the class $\mathsf{AC}^0(\oplus)$, which extends AC^0 by allowing also (unbounded fan-in) parity gates, and showed that this class cannot compute the majority or mod$p$ functions. Most recently, Williams [54] separated the class ACC^0, in which the circuit is further allowed to use arbitrary mod$p$ gates, from the class NEXP of non-deterministic exponential-time computations (see also the recent exciting sequence of works [18, 53, 16, 17, 42]).

Despite these results, we are still far from understanding the power of constant-depth circuits. For example, it is easy to construct an $\mathsf{AC}^0(\oplus)$ circuit for computing the inner product function: simply take the parity of the respective point-wise products. On the other hand, if we do not allow parity gates, then it is easy to show a lower bound. A natural question that arises is whether a similar lower bound holds if we augment the AC^0 circuit with a single layer of parity gates immediately after the input layer. The resulting circuit class is called an AC^0 of parities (and is sometimes denoted by ${\mathsf{AC}^0}_{\oplus}$). Recently, there has been growing interest in whether this circuit class can compute the inner product function [34, 45, 48, 3, 20, 19, 9, 22].

Interestingly, an exponential lower bound for the inner product function *is* known [31] for the special case in which the AC^0 circuit has depth 2.[1] Namely, a DNF of parities, denoted by $\mathsf{DNF}_{\oplus}$. For depth 3 circuits (on top of the parity layer), only a relatively weak (quadratic) lower bound is known [19]. Lastly, for general ${\mathsf{AC}^0}_{\oplus}$ circuits, an exponential lower bound is known [22] only for the very restricted case in which the number of parity gates is linear.[2]

Even for the case of DNFs, the lower bound arising from the work of Jackson [31] only rules out an (almost) *exact* $\mathsf{DNF}_{\oplus}$ for computing the inner product function. Thus, a question (posed explicitly by Cohen and Shinkar [20]) that seems just beyond the reach of current techniques is:

> *"Does there exist a small DNF of parities that approximates the inner product function?"*

We refer to the assumption that a positive answer holds for this question as the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis (in the actual theorem statements below we use a quantitatively precise version of the assumption). In this work, we study the ramifications of the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis, with the belief that these results develop our understanding of the hypothesis or could even bring us closer to the eventual goal of *refuting* it.

Recently, in a work of Huang et al. [30], it was shown that a positive answer to the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis implies a small ${\mathsf{AC}^0}_{\oplus}$ for the inner product function (in the worst case). Still, proving or refuting the hypothesis remains beyond the grasp of current techniques.

1.1 Our Results

We show that a positive answer to the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis implies (unexpected) efficient interactive (Arthur-Merlin) protocols for a large class of problems (in different models to be described below). We note that the quantitative parameters of the resulting protocols seem to be far more efficient than expected. Thus, these results fall in line with our belief that the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis is false. Moreover, these results also form a new approach for *refuting* the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis through Arthur-Merlin lower bounds, in the sense that progress in finding Arthur-Merlin lower bounds can be applied, using our results, to refute the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_{\oplus}$ hypothesis. While finding lower bounds for Arthur-Merlin protocols is a notoriously difficult problem (e.g., in communication complexity), all of our protocols go through efficient *Holographic-Interactive protocols*, where Arthur-Merlin lower bounds *are* known [28].

---

[1] This bound was tightened by Cohen and Shinkar [20], who gave a lower bound that exactly matches the known upper bound.

[2] In fact, their result holds for the more general case in which an arbitrary (i.e., not necessarily linear) preprocessing step is done first on the two parts of the input separately.

The models that we consider are "Arthur-Merlin" variants of the standard Data Streaming and the Communication Complexity models. In order to describe these variants, we first recall the standard definitions of Data Streaming and Communication Complexity models and then explain how they are extended to Arthur-Merlin variants, by giving the relevant parties access to an all-powerful (but untrusted) prover.

Recall that in the standard Data Streaming Model (popularized by [4]), a bounded space algorithm is required to compute a certain function of the inputs by using the least amount of space. The algorithm gets the input bits as a sequence of bits (stream), in the sense that after seeing a bit in the sequence, the algorithm no longer has access to the bits that preceded it (unless these were stored in its memory). In the standard Communication Complexity Model [55], there are two parties, called Alice and Bob, who are trying to evaluate a function f on their joint input. That is, Alice and Bob are given inputs x and y, respectively, and need to jointly compute the value $f(x, y)$, while transmitting the least amount of bits.

We focus on the Arthur-Merlin (AM) variant of these models, where the parties are also assisted by an untrusted prover, often referred to as Merlin, who sees all inputs (and has unlimited computational resources). The parties are allowed to make a short *public coin* interaction with Merlin, *before* (deterministically) running the standard protocol. The interaction is of the AM (Arthur-Merlin) type in the sense that the messages to Merlin consist of only fresh random coins, and in particular, do not depend on the input bits nor on previous messages that were exchanged. Beyond the coins that were revealed to Merlin in the interaction, the parties are not allowed to toss any additional coins. Throughout this work we use the notation $\mathsf{AM}[k]$ to refer to AM protocols with k messages exchanged between the parties.

1.1.1 The AM Data Streaming Model

In the AM Data Streaming Model [21, 13, 27, 11, 12, 14, 52, 15], we allow the bounded space algorithm processing the stream, to interact with the untrusted prover Merlin, who sees the entire input (and is not space bounded). Many of these works differ in the exact form of the interaction. For example, does the small-space verifier get full access to messages sent by the prover, or merely streaming access? In this work we consider the following natural model, which we refer to as the $\mathsf{AM}[k]$ Data Streaming model:

1. In the first phase, the verifier engages in a k-message public-coin interactive protocol with the prover (starting with a verifier message). At the end of this phase the verifier holds a transcript τ.
2. In the second phase, the verifier is allowed to process the input stream in a bit-by-bit manner. The verifier's computation in this phase is allowed to depend on the transcript τ that it saw. We emphasize that the verifier in this phase is deterministic.[3]
3. After processing the stream, the verifier decides whether to accept or reject.

As usual, we require that there is a strategy for Merlin to convince the streaming verifier to accept true statements, but the verifier rejects any false statements (with high probability) even if Merlin cheats. Naturally, we require the space complexity of the verifier to be small and the communication with the prover to be short as well (since otherwise Merlin can provide the entire input!).

[3] The verifier could in principle toss additional coins in the first phase to be used in the second phase, but we count this as an additional message. This is motivated by the definition of the classical complexity class AM which does not allow Arthur additional coin tosses after seeing Merlin's message.

As our first result, assuming the existence of a small DNF of parities for the inner product, we construct (multi-round) AM Data Streaming protocol for *any* function f that can be computed by a low-degree polynomial (over $\mathbb{GF}(2)$).

▶ **Theorem 1** (Informally stated (see Theorem 9)). *Assume that there exists a* DNF *of parities of size S, that computes the inner product function on $\frac{5}{6}+\epsilon$ fraction of the inputs, for some constant $\epsilon > 0$. Then, there exists an* AM$[2d]$ *Data Streaming protocol with $\widetilde{O}(d) \cdot \log(S)$ proof length, space complexity and randomness complexity, for every degree d polynomial over $\mathbb{GF}(2)$.*

When S is polynomial and d is (super) constant, the protocol has poly-logarithmic proof and space complexities. This should be contrasted with approaches based on (super) constant-round versions of the celebrated sumcheck protocol [40], which have polynomial proof-length.

We also emphasize that (as usual in this context) here and throughout this work, we do not consider the *computational complexity* of the verifier and only focus on the space and communication complexities.

Application: A Streaming Protocol for Counting Triangles Mod 2. We also point out an interesting implication of Theorem 1 to a variant of the well studied Triangle-Count problem. In the Triangle-Count problem, a streaming algorithm is required to count (or sometimes just approximate) the number of triangles (i.e., cliques of three vertices) in an undirected (simple) graph. A large body of work has studied this problem in the streaming context in general [7, 33, 10, 39, 36, 32, 41, 8, 35], and in particular when the streaming algorithm is assisted by a prover [12, 52, 15]. There are two main variants of the Triangle-Count problem, which differ in the exact form that the input is given to the streaming algorithm. In the first variant, studied in [7, 10, 39, 41, 35], the edges are given in an adjacency-list format. Namely, first, the edges connected to the first vertex appear in the stream, then the edges that are connected to the second vertex, and so on. In the second variant (also referred to as the dynamic updates variant), studied in [7, 33, 10, 36, 32, 41, 8], the stream consists of dynamic additions (and sometime also deletions) of edges, in an arbitrary order.

We consider a variant of the Triangle-Count problem, denoted by $\bigoplus$Triangle, where the goal is to compute the *parity* of the number of triangles in the graph. We consider in which the graph is given as a stream of its edges, where each edge appears in the stream exactly once. We note that the MA complexity[4] of $\bigoplus$Triangle is well understood: for every proof length p and verifier space complexity s, it holds that $s \cdot p = \Omega(n^2)$ [12, 52].[5] Also, a matching quadratic upper bound is known for (almost) any combination of $s \cdot p = \tilde{O}(n^2)$ [52, 15]. On the other hand, this problem has no known (non-trivial) upper or lower bounds in the AM setting.

Assuming the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_\oplus$ hypothesis, we show an efficient AM protocol for $\bigoplus$Triangle. Our protocol has space complexity and proof length that are poly-logarithmic in the circuit size (regardless of the specific order of the edges in the stream).

▶ **Theorem 2** (AM Streaming for $\bigoplus$Triangle, informally stated (see Theorem 11)). *Assume that there exists a* DNF *of parities of size S that computes the inner product function on $\frac{5}{6}+\epsilon$ fraction of the inputs, for some constant $\epsilon > 0$. Then, there exists an* AM$[2]$ *Data Streaming protocol for $\bigoplus$*Triangle *with* polylog$\left(S\left(n^3\right)\right)$ *proof length and space complexity.*

[4] Loosely speaking, in an MA model, first the prover sends a proof message. Then, the verifier gets the input as a stream, and conducts a (randomized) streaming computation.

[5] The lower bound is not stated explicitly for this problem, but follows from the fact that it holds for the case that the graph is promised to contain exactly one triangle or be triangle-free.

Indeed, assuming that S is polynomial, the protocol of Theorem 2 has poly-logarithmic proof length and space complexity.

1.1.2 AM Communication Complexity

We next describe our results in the (AM) Communication Complexity model. In the AM Communication Complexity Model [37, 1, 24, 38, 25], Alice and Bob are allowed to also conduct a public-coin interaction with the prover Merlin, who sees both of their inputs, but is non trustworthy. The parties communicate using a broadcast channel, namely, each of the parties is exposed to all the messages sent by Merlin, and all the random coins tossed by Alice and Bob. For sake of simplicity, we can assume that Alice and Bob do *not* interact, since Merlin can simply provide all messages that they would have exchanged had they interacted (and the two parties can check that the communication is consistent with what they would have sent). As above, we require that Merlin will convince *both* Alice and Bob of the correctness of true statements, but no matter what Merlin does, with high probability either Alice or Bob will reject false statements.

It is not hard to show that any Data Streaming protocol can be transformed into a Communication Complexity protocol, for the same problem, as follows: Alice starts running the data streaming algorithm until the algorithm finishes processing her portion of the input (i.e., at the midpoint). She then transmits to Bob her memory state. Bob continues the emulation using his portion of the input. The communication complexity of the resulting protocol is therefore at most the space complexity of the streaming algorithm.

Thus, Theorem 1 immediately implies an AM communication complexity protocol for low-degree polynomials as well. Interestingly, however, we are able to achieve significantly better parameters by constructing an AM Communication Complexity protocol directly. In particular, we construct a one-round protocol, which can also be extended to a protocol for any function that is decidable by an $\mathsf{AC}^0(\oplus)$ circuit. Lastly, we also note that while the protocol in Theorem 1 depends on the degree of the polynomial, the protocol in Theorem 3 depends only on the number of monomials, and therefore can also be applied to high-degree, but sparse, polynomials.

▶ **Theorem 3** (Informally stated (see Theorem 7 and Corollary 8)). *Assume that there exists a* DNF *of parities of size S that computes the inner product function on $\frac{5}{6}+\epsilon$ fraction of the inputs, for some constant $\epsilon > 0$. Then, there exist:*

- *An* AM[2] *Communication Complexity protocol with $O\left(\log\left(S(2N)\right)\right)$ communication complexity, for every function f that can be expressed as a polynomial (over $\mathbb{GF}(2)$) with N monomials.*
  In particular, if f is a degree d polynomial over $2n$ input bits, the AM[2] *protocol has communication complexity $O\left(\log\left(S\left(2\cdot(2n)^d\right)\right)\right)$.*
- *An* AM[2] *Communication Complexity protocol with $O\left(\log\left(S\left(2^{\mathrm{polylog}(T)}\right)\right)\right)$ communication complexity, for any function that is decidable by an $\mathsf{AC}^0(\oplus)$ circuit of size T.*

Note that the protocols in Theorem 3 are 2-message protocols, whereas the protocol in Theorem 1 require a large number of rounds of interaction. One could potentially reduce the number of rounds using (a suitable variant of) the round collapse theorem [6] (see also [46, Lemma 4.6]). However, we emphasize that Theorem 3 gives significantly better parameters than round collapsing the protocol of Theorem 1. For example, if $S = \mathrm{poly}(n)$, by applying a round collapse to our data streaming results, we get an AM[2] Communicating Complexity protocol with $O\left(\log^d n\right)$ communication complexity for degree d polynomials. In contrast,

our explicit protocol of Theorem 3 has a *linear* (rather than exponential) dependence on the degree d. This improvement allows us to extend the explicit Communication Complexity protocol for low degree (or sparse) polynomials, also for any function that is decidable by an $\mathsf{AC}^0(\oplus)$ circuit. Interestingly, we do not know how to obtain a non-trivial result of the same flavor from the protocol in Theorem 1.

1.2 Technical Overview

In this section, we present the main methods and techniques that used in our work. For the full details and proofs, please refer to the technical sections in the full version [47].

Our main technical step is to construct, assuming that the $\mathsf{IP}_2\widetilde{\in}\mathsf{DNF}_\oplus$ hypothesis holds, a special type of proof-system for computing the inner product function, called a *Holographic Interactive Proof* (HIP) - a notion introduced in the work of Gur and Rothblum [28] (inspired by a similar model for PCPs, introduced by Babai et al. [5]). An HIP is defined similarly to a standard interactive proof, except that the verifier, rather than being given access to the main input explicitly, is given oracle access to an *encoding* of the input. The hope is that the redundancy provided by the encoding will allow the verifier to run in *sub-linear* time. Hence, the main complexity resources that we focus on are the *query complexity*, which is the number of bits that the verifier reads from the encoding, and the *communication complexity*, which is the total number of bits exchanged with the prover. We focus specifically on an AM[2] variant, where the verifier first sends random coins r to the prover, who responds with a message π, called the proof. The verifier then decides deterministically, based on the input queries, random string r and proof π, whether to accept or reject.

Let us assume therefore that there exists a DNF of parities C of size S that approximates the inner product function. We use C to design an AM[2] HIP for verifying inner product claims. The input encoding that we will use in the HIP corresponds to the parity layer of the circuit C, and is therefore a *linear* function. This point is crucial for our results.

1.2.1 An AM[2]-HIP for Inner Product Claims

As our first step, we construct a simple HIP for verifying that the inner product of two strings is equal to 1 and which only works for *most* inputs. This falls short of our eventual goal which is to check general inner products and over *worst-case* inputs. Nevertheless, we present this HIP as it will serve as an important ingredient in our construction.

Step 1: Verifying one-sided claims, on the average. Recall that C is a DNF of parities that approximates the inner product function. A simple one-round HIP protocol for verifying whether $f(x) = 1$ on a given input x can be established as follows: the prover sends an index of a satisfied clause (such an index exists if and only if $f(x) = 1$), and the verifier checks whether the clause is indeed satisfied, by reading the bits in the clause from the input's encoding. Note that the proof-system is holographic as the verifier reads each bit in the clause by making a single query to the output of the parity layer. The communication is $\log(S)$ and the query complexity is bounded by the maximal width of the clauses.

Since we seek small query complexity, we would like to ensure that the DNF has small width. To do so we observe that each clause in a DNF of parities can be viewed as a system of linear equations. Also, note that with probability at most $\frac{1}{2^r}$, a random input satisfies a linear system with rank at least r. Therefore, a natural idea is to remove all of the wide clauses. When doing so one should first make sure that the equations forming the clause are linearly independent, which can be easily done (by choosing a maximal set of linearly independent

equations). Thus, after eliminating linear dependencies, we remove all clauses with width $\Omega(\log S)$. Since we only removed clauses, the new circuit disagrees with $f(x)$ only if x satisfies one of the removed clauses. Since we only removed clauses of rank greater than $O(\log S)$, by the union bound and setting the constant in the big-O to be large enough, the probability that an input x satisfies one of the removed clauses is at most $\frac{S}{2^{O(\log S)}} = \frac{1}{\text{poly}(S)} = o(1)$.

Overall we have constructed an HIP that can verify whether an inner product of two strings is 1 on most inputs, with $O(\log S)$ proof length, and $O(\log S)$ query complexity. As noted before, our next step is to convert this protocol – which works in the average case – into a protocol that can verify *any* inner product claim.

Self-correction of the inner product function. As an initial observation, we observe that the self-correction property of linear functions can be extended also to the inner product function (this can also be viewed as a special case of locally decoding the Reed-Muller code over $\mathbb{GF}(2)$, see [26]). For any input strings x, y, and vectors u, v' which are taken at random, it holds that

$$\langle x, y\rangle = \langle x \oplus u, y \oplus v\rangle \oplus \langle x \oplus u, v\rangle \oplus \langle u, y \oplus v\rangle \oplus \langle u, v\rangle, \tag{1}$$

where $\langle a, b\rangle$ denotes the inner product of strings $a, b \in \{0,1\}^n$. Note that the terms on the right-hand side of Equation (1), are inner products over different (correlated) random inputs. Also, recall that the "simple" protocol that was described previously, can verify inner product claims about random inputs with high probability over the inputs. Thus, at first glance it may seem sufficient to use *Equation* (1), and verify the random claims using our average-case protocol. Unfortunately, by moving to claims over inner products of random inputs we will also need the ability to verify whether an inner product is 0, while so far we only have an HIP for "1-claims". Therefore, instead of using Equation (1) directly, we present a generic compiler that extends the self-correction property of Equation (1) also to the case where there is a protocol that can only verify *most* of the 1-claims (a similar idea was used also in the works of Shaltiel and Umans [49, 50]). In this compiler, we rely on the fact that in our HIP the prover can't convince the verifier to accept a false 1-claim (with high probability over the inputs). For simplicity, we outline this compiler with respect to protocols for the inner product function but in the technical sections, we extend this argument to any *homogeneous multilinear mapping*.

Step 2: Self-correction with one-sided errors. In order to use Equation (1), we need to verify also the 0-claims on the right-hand side of Equation (1). Observe, that since Equation (1) gives inner product claims of (individually) random inputs, then, in expectation, about half will be 0's and half will be 1's.

Thus, since (with high probability) a cheating prover cannot lie on false 1-claim, it will likely have to generate false 0-claims and therefore skew the distribution of 0 vs. 1 claims. In order to detect this, we simply repeat the experiment sufficiently many times (using independent coin tosses) and checking the empirical average value of the prover's claims. To sum up, given a ground protocol that works only on most 1-claims, the compiler produces the following protocol: the verifier uses Equation (1) several times, each time with fresh random strings. At each iteration, the prover sends the values of the random inner products on the right-hand side of Equation (1), while having the verifier check only the 1-claims, by using the ground protocol, and blindly accepting the 0-claims. At the end of the interaction, the verifier checks whether the average value of all the prover's claims is close enough to the expectation of the inner product function. If the average is close enough, the verifier infers that the prover is honest. Otherwise, the verifier infers that the prover lies, and thus it rejects.

Lastly, in order to reduce the randomness complexity to $O(\log n)$ randomness complexity, we use a standard technique, due to Newman [43], for reducing the randomness complexity (using non-uniformity). We show that this technique works also in the context of AM-HIPs.

An alternate approach. We find it also instructive to describe another approach for dealing with the 0-claims in Equation (1), and explain the reason we decided not to use it. The idea here is to show a random self-reduction from a 0-claim to a 1-claim. This can be done by embedding the input strings x and y into longer random string strings x' and y', while ensuring that the $\langle x', y' \rangle = 1 \oplus \langle x, y \rangle$.

The reason we decided not to follows this approach is that it changes the input size. In particular, it would mean that the verifier in the HIP would need to access a different linear transformation then that in the bottom layer of the DNF.

1.2.2 From HIP to Communication Complexity (Proving Theorem 3)

Our key idea in proving to construct an AM Communication Complexity protocol for sparse polynomials is the observation that a polynomial can be viewed as a linear combination of its monomials. In the communication complexity setting, each monomial is a product between a subset of Alice's input bits, and a subset of Bob's input bits. Thus, we can view the evaluation of the polynomial $f(x, y) = \sum_{\text{monomial } (\alpha,\beta)} x_\alpha \cdot y_\beta$, where $x_\alpha = \prod_{i \in \alpha} x_i$ and $y_\beta = \prod_{j \in \beta} y_j$, as an inner product between the strings (x_α) and (y_β).

Thus, in order to solve the problem, it suffices to construct an AM Communication Complexity protocol for computing the inner product function. Such a protocol follows easily from our HIP for inner products - since each query that the HIP verifier makes, is a *linear* query to the joint input (x, y), it can be emulated by having Alice and Bob compute and share their individual contributions.

The second part of Theorem 3 now follows easily by observing that every degree d polynomial over $\mathbb{GF}(2)$ can have at most n^d monomials, and by applying the polynomial approximation method of Razborov and Smolensky [44, 51] (where the choice of the random polynomial can be made by the verifier as part of its first step in the protocol).

1.2.3 From HIP to Data Streaming (Proving Theorem 1)

Unfortunately, our approach for computing sparse polynomials that worked in the communication complexity setting, fails in the *streaming* setting. The issue is that each monomial consists of a product of *multiple* input bits. Therefore, the polynomial's monomials induce an inner product between a coefficient vector, and a *tensor* of the input, rather than the input in its basic form. While it is relatively easy to make queries to an encoding of the input by a streaming verifier, it is not clear at all how to make queries to an encoding of a *tensor* of the input.

Nevertheless, our starting point is the above observation that a polynomial can be expressed as a certain inner product. In more details, a degree d polynomial $\mathcal{P} : \{0,1\}^n \to \{0,1\}$ (over the field $\mathbb{GF}(2)$) can be viewed as a linear combination of its monomials, each of which is a product between a coefficient and a product of d input bits. Therefore, there exists a function $\mathsf{Coef}_{\mathcal{P}} : [n]^d \to \{0,1\}$ that depends only on $\mathcal{P}$, such that:

$$\mathcal{P}(x) = \bigoplus_{j_1,\ldots,j_d \in [n]} x_{j_1} \cdot x_{j_2} \cdot \cdots \cdot x_{j_d} \cdot \mathsf{Coef}_{\mathcal{P}}(j_1, \ldots, j_d). \tag{2}$$

The basic idea of the protocol is to iteratively use the HIP protocol for inner product claims (henceforth, the ground HIP protocol), to gradually reduce a claim about the right-hand side of Equation (2), to claims that don't depend on the inputs - that is, claims that depend only the structure of the specific code that the HIP uses, and the structure of the polynomial $\mathcal{P}$. Since the resulting claims do not depend on the input, the verifier will be able to check them without additional communication or queries.

Inspired by the celebrated sumcheck protocol of Lund *et al.* [40], we construct a d-round AM-HIP protocol, so that in the i-th round we reduce a set of claims over $d-(i-1)$ input variables, to a set of claims that depend on only $d-i$ input variables. The i-th round starts with claims of the form:

$$\bigoplus_{j_1,\ldots,j_d\in[n]} \left(\hat{\beta}^{(i-1)}\left(j_1,\ldots,j_{i-1}\right)\cdot x_{j_i}\cdot x_{j_{i+1}}\cdots x_{j_d}\cdot \mathsf{Coef}_{\mathcal{P}}(j_1,\ldots,j_d)\right) = b^{(i-1)}, \tag{3}$$

where $\hat{\beta}^{(i-1)}$ is a function that depends only on the structure of the linear code that the base HIP protocol uses. Our goal is to end the round with multiple claims of the form:

$$\bigoplus_{j_1,\ldots,j_d\in[n]} \hat{\beta}^{(i)}(j_1,\ldots,j_i)\cdot x_{j_{i+1}}\cdots x_{j_d}\cdot \mathsf{Coef}_{\mathcal{P}}(j_1,\ldots,j_d) = b^{(i)}.$$

Observe that by changing the order of summation in Equation (3), we can rewrite each claim as:

$$\bigoplus_{j_i\in[n]} x_{j_i}\cdot\left(\bigoplus_{\substack{j_1,\ldots,j_{i-1}\in[n],\\ j_{i+1},\ldots,j_d\in[n]}} \left(\hat{\beta}^{(i-1)}\left(j_1,\ldots,j_{i-1}\right)\cdot x_{j_{i+1}}\cdots x_{j_d}\cdot \mathsf{Coef}_{\mathcal{P}}(j_1,\ldots,j_d)\right)\right) = b^{(i-1)}. \tag{4}$$

The claims in Equation (4) are an inner product between x and the truth table of a function that depends on only $d-i$ inputs variables. Thus, by applying the ground HIP protocol, we get multiple claims on the encoding of x, and multiple claims on the encoding of the truth table of a function that depends on $d-i$ variables. The claims on the encoding of x can be verified using the HIP verifier's oracle queries. Regarding the second class of queries, since the code is linear, the t-th claim is of the form of

$$\bigoplus_{j_i\in[n]} \gamma_{t,z}(j_i)\cdot \bigoplus_{\substack{j_1,\ldots,j_{i-1}\in[n],\\ j_{i+1},\ldots,j_d\in[n]}} \left(\hat{\beta}^{(i-1)}\left(j_1,\ldots,j_{i-1}\right)\cdot x_{j_{i+1}}\cdots x_{j_d}\cdot \mathsf{Coef}_{\mathcal{P}}(j_1,\ldots,j_d)\right) = b^{(i)}_{t,z}.$$

where the $(\gamma_{t,z})$'s correspond to the coefficients of the base code T. By changing the order of summation again, and defining $\hat{\beta}^{(i)}_{z,t}\left(j_1,\ldots,j_i\right) = \gamma_{t,z}(j_i)\cdot\hat{\beta}^{(i-1)}_{t}\left(j_1,\ldots,j_{i-1}\right)$ we get claims of the form:

$$\bigoplus_{j_1,\ldots j_d\in[n]} \left(\hat{\beta}^{(i)}_{t,z}\left(j_1,\ldots,j_i\right)\cdot x_{j_{i+1}}\cdots x_{j_d}\cdot \mathsf{Coef}_{\mathcal{P}}(j_1,\ldots,j_d)\right) = b^{(i)}_{t,z},$$

where the $\hat{\beta}^{(i)}_{t,z}\left(j_1,\ldots,j_i\right)$ don't depend on the input variables. Note that we got new claims that depend on $d-i$ input variables, as required.

Avoiding a complexity blowup. Although the above idea seems promising, we have one additional issue to deal with. In contrast to the traditional sumcheck protocol which generates a single claim in the end of each round, our ground HIP protocol produces multiple claims.[6]

[6] Recall that the query complexity is roughly logarithmic in the size of our DNF of parities.

As a result, the number of times we need to use the ground HIP protocol grows by a at least a constant factor in each round, and overall becomes (at least) exponential in d. In order to reduce the dependence on d to *linear*, at the beginning of each round we combine claims together by taking random linear combinations. This method lets us preserve the number of queries after each round, and thus achieve a linear dependence on d.

On the approximation factor. The (roughly) $\frac{5}{6}$ approximation factor required in all of our results, stems from the use of Equation (1). Recall that the verifier needs to check all the 1-claims on the right-hand side of Equation (1) with a success probability greater than $\frac{1}{2}$. Leveraging the fact that one of the terms on the right-hand side of Equation (1) is independent of the input strings, we only have three terms to check. As a result, we must have a circuit that computes all the three terms correctly with probability greater than $\frac{1}{2}$. By union bounding over these three terms, we get that the circuit must compute a random input incorrectly with probability at most $\frac{1}{6}$, which sets the approximation limitation to $\frac{5}{6}$. A potential approach for improving the approximation factor is to rely on locally list, and we leave this possibility to future work.

1.2.4 Counting Triangles Mod 2 (Theorem 2)

Lastly, we give the outline of the streaming protocol for the $\bigoplus$Triangle problem. To do so we leverage the fact that $\bigoplus$Triangle can be expressed as a degree 3 polynomial over $\mathbb{GF}(2)$ and apply the streaming protocol of Theorem 1.

In more detail, let $\{I_{(u,v)}\}_{u,v}$ be a set of indicator variables where $I_{(u,v)}$ is 1 if and only if the edge (u,v) appears in the graph. We can express the parity of the number of triangles in the graph, by evaluating the following degree 3 polynomial:

$$\mathcal{P}_{\oplus\text{TRI}}\left(I_{e_1},\ldots,I_{e_{n^2}}\right) = \bigoplus_{v<u<w\in[n]} I_{(v,u)}\cdot I_{(u,w)}\cdot I_{(w,v)}.$$

Thus, by applying our streaming protocol for low degree polynomials from Theorem 1, we derive an AM[6] Data streaming protocol for $\bigoplus$Triangle. Lastly, in order to derive a one-round protocol, we use the round collapsing technique of Babai and Moran [6] for reducing the number of rounds in public coin interactions.

2 Our Results

In the following section we present the formal version of our results. We start by introducing some notations. Then, we present our results in the AM Holographic Interactive Proof, AM Communication Complexity, and AM Data Streaming models, in that order. The full proofs of our results along with the formal definitions of the different models are available in the full version of this paper [47].

2.1 Preliminaries

By AM[k]-DS, AM[k]-CC and AM[k]-HIP we denote the k-message AM Data Streaming, AM Communication Complexity and AM Holographic Interactive protocols, respectively.

By $\mathsf{IP}_2(x,y)$ we denote the inner product function (over the field $\mathbb{GF}(2)$), that is, $\mathsf{IP}_2(x,y) = \oplus_{i\in[n]} x_i\cdot y_i$. And by L_{IP} we denote its corresponding language:

▶ **Definition 4.** (L_{IP} language).

$$L_{\mathsf{IP}} \stackrel{def}{=} \Big\{(x,y,b)\in\{0,1\}^n\times\{0,1\}^n\times\{0,1\} \;:\; \mathsf{IP}_2(x,y)=b\Big\}.$$

DNF ∘ T Circuits

For a linear transformation T, we denote by $\mathsf{DNF} \circ \mathsf{T}$ the circuit that is a composition of some circuit that computes the transformation T, with a DNF circuit (disjunction of clauses). We note that since T is linear, these circuits are a particular type[7] of a DNF of parties, where the parity gates are only allowed to compute the function T. We use this notation to point out the connection between the specific function the parity layer of the circuit in the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_\oplus$ hypothesis computes and the linear code our Holographic verifier makes queries to.

2.2 Holographic Interactive Proof for Inner Product Claims

As described in Section 1.2, all our protocols are based on a special type of proof-system for computing the inner product function, called a *Holographic Interactive Proof*. Assuming the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_\oplus$ hypothesis, our first step in our work is to construct an efficient (non-uniform) AM[2]-HIP protocol for checking inner product claims.

▶ **Lemma 5.** (AM[2]-HIP for L_{IP}). *Fix an integer n, and a parameter $\epsilon \in (0, 1/6]$. Let $\mathsf{T} : \{0,1\}^{2n} \to \{0,1\}^{n'}$ be some linear code. Suppose there exists a $\mathsf{DNF} \circ \mathsf{T}$ circuit C of size S that computes $\mathsf{IP}_2(x, y)$ on at least $\frac{5}{6} + \epsilon$ fraction of the inputs. Then, there exists an AM[2]-HIP protocol for L_{IP}, with proof length $\log(S) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$, randomness complexity $O(\log n)$ and $\log(S) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$ queries to the bits of $\mathsf{T}(x, y)$.*

Using standard transformations from AM-HIP to AM-CC and AM-DS we also derive AM-CC and AM-DS protocols for the inner product function.

▶ **Corollary 6.** (AM[2]-DS and AM[2]-CC for L_{IP}). *Let $n, n' \in \mathbb{N}$, and $\epsilon \in (0, 1/6]$. Let $\mathsf{T} : \{0,1\}^{2n} \to \{0,1\}^{n'}$ be some linear code. Suppose there exists a $\mathsf{DNF} \circ \mathsf{T}$ circuit C of size S that computes IP_2 on at least a $\frac{5}{6} + \epsilon$ fraction of the inputs. Then,*

1. *There exists an AM[2]-DS protocol for L_{IP}, with proof length $\log(S) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$, randomness complexity $O(\log n)$ and verifier space complexity $\log(S) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$.*
2. *There exists an AM[2]-CC protocol for the function $\mathsf{IP}_2(x, y)$ with $\log(S) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$ communication complexity.*

2.3 An AM[2] Communication Complexity Protocol

Our next results focus on the AM Communication Complexity model. Assuming the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_\oplus$ hypothesis, we first construct an efficient AM[2]-CC protocol for low degree polynomials.

▶ **Theorem 7.** *Fix integers n, N and a parameter $\epsilon \in (0, 1/6]$. Let $\mathsf{T} : \{0,1\}^{2N} \to \{0,1\}^{n'}$ be some linear code. Suppose there exists a $\mathsf{DNF} \circ \mathsf{T}$ circuit C of size $S = S(2N)$ that computes the function $\mathsf{IP}_2(x, y)$ on at least a $\frac{5}{6} + \epsilon$ of the N-bit length inputs. Then, for any polynomial $\mathcal{P} : \{0,1\}^n \to \{0,1\}$ with N monomials, and for any $b \in \{0,1\}$, there exists an AM[2]-CC protocol for the function $\mathcal{P}_b(x, y) \stackrel{def}{=} \begin{Bmatrix} 1 & \mathcal{P}(x,y) = b \\ 0 & \text{o/w} \end{Bmatrix}$ with $\log\left(S\left(2N\right)\right) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$ communication complexity, where Merlin (the prover) gets the inputs $x, y \in \{0,1\}^n$, Alice gets x and Bob gets y.*

[7] Namely, DNF circuits with an additional layer of parity (XOR) gates which can be applied only directly on the input gates.

Then, relying on the celebrated works of Razborov and Smolensky [44, 51] which showed a general technique to approximate $\mathsf{AC}^0(\oplus)$ circuit by a distribution of randomized low degree polynomials, we also construct an $\mathsf{AM}[2]$-DS protocol for every language that is decidable by an $\mathsf{AC}^0(\oplus)$ circuit.

▶ **Corollary 8.** *Fix an integer n, and let $\epsilon \in (0, 1/6]$. Suppose that for any k there exists a linear transformation $\mathsf{T} : \{0,1\}^{2k} \to \{0,1\}^{\ell(k)}$ and a $\mathsf{DNF} \circ \mathsf{T}$ circuit C of size $S_0(2k)$ that computes the function $\mathsf{IP}_2(x,y)$ on at least a $\frac{5}{6} + \epsilon$ of the k-bit length inputs. Then, there exists a constant c such that for any function $f : \{0,1\}^{2n} \to \{0,1\}$ that is computed by an $\mathsf{AC}^0(\oplus)$ circuit of size S and depth $d \geq 2$, there exists an $\mathsf{AM}[2]$-CC protocol for f with $\log\left(S_0(2^{(c \cdot \log n \cdot \log^d S)})\right) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$ communication complexity.*

2.4 An AM[$2d$] Streaming Protocol

Lastly, we focus on the Data Streaming model. Our main result in the streaming model is the construction of a (multi-round) AM Data Streaming protocol for low degree polynomials, assuming that the $\mathsf{IP}_2 \widetilde{\in} \mathsf{DNF}_\oplus$ hypothesis holds.

▶ **Theorem 9.** *Fix an integer n, and let $\epsilon \in (0, 1/6]$. Let $\mathsf{T} : \{0,1\}^{2n} \to \{0,1\}^{n'}$ be some linear code. Suppose there exists a $\mathsf{DNF} \circ \mathsf{T}$ circuit C of size S that computes the function $\mathsf{IP}_2(x,y)$ on at least a $\frac{5}{6} + \epsilon$ fraction of the n-bit length inputs. Then, for any d degree polynomial $\mathcal{P} : \{0,1\}^n \to \{0,1\}$, and for any $b \in \{0,1\}$, there exists an $\mathsf{AM}[2d]$-DS protocol for the language $L \stackrel{def}{=} \{x \in \{0,1\}^n \mid \mathcal{P}(x) = b\}$ with $\log(S) \cdot \widetilde{O}\left(\frac{d}{\epsilon^3}\right)$ randomness complexity, proof length and space complexity.*

We also point out an interesting implication of Theorem 9 to a variant of the well studied $\mathsf{Triangle\text{-}Count}$ problem, called the $\bigoplus \mathsf{Triangle}$. In the $\bigoplus \mathsf{Triangle}$ the verifier is required to count the *parity* of the number of triangles (i.e. cliques with three vertices) in an undirected (simple) graph $G = (V, E)$.

▶ **Definition 10.** *Let $G = (V, E)$ be an undirected simple graph such that $V \subseteq [n]$ and $E \subseteq [n] \times [n]$. In the $\bigoplus \mathsf{Triangle}$ problem, the edges in E are given as a stream in some arbitrary order, where each edge appears in the stream exactly once. The goal is to output the parity of the number of triangles (i.e. cliques of size 3) in G.*

Our last result shows that the existence of a sufficiently small $\mathsf{DNF} \circ \mathsf{T}$ circuit that approximates the inner product function, yields an efficient $\mathsf{AM}[2]$-DS protocol for $\bigoplus \mathsf{Triangle}$.

▶ **Theorem 11.** *Fix an integer n and a parameter $\epsilon \in (0, 1/6]$. Let $\mathsf{T} : \{0,1\}^{n^3} \to \{0,1\}^{n'}$ be a linear code. Suppose there exists a $\mathsf{DNF} \circ \mathsf{T}$ circuit C of size S that computes the function $\mathsf{IP}_2(x,y)$ on at least a $\frac{5}{6} + \epsilon$ fraction of the n^3-bit length inputs. Then, there exists an $\mathsf{AM}[2]$-DS protocol for $\bigoplus \mathsf{Triangle}$ with $O(\log n)$ randomness, and $\log^3(S) \cdot \widetilde{O}\left(\frac{1}{\epsilon^3}\right)$ verifier space complexity and proof length.*

References

[1] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1(1):2:1–2:54, 2009. doi:10.1145/1490270.1490272.

[2] Miklós Ajtai. $\sum^1_1$-formulae on finite structures. *Ann. Pure Appl. Log.*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.

[3] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $AC^0 \circ mod_2$. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 251–260. ACM, 2014. doi:10.1145/2554797.2554821.

[4] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.

[5] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31. ACM, 1991. doi:10.1145/103418.103428.

[6] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988. doi:10.1016/0022-0000(88)90028-1.

[7] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In David Eppstein, editor, *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 623–632. ACM/SIAM, 2002. URL: http://dl.acm.org/citation.cfm?id=545381.545464.

[8] Suman K. Bera and Amit Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.11.

[9] Mark Bun, Robin Kothari, and Justin Thaler. Quantum algorithms and approximating polynomials for composed functions with shared inputs, 2020. arXiv:1809.02254.

[10] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In Stijn Vansummeren, editor, *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 253–262. ACM, 2006. doi:10.1145/1142351.1142388.

[11] Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 687–706. SIAM, 2014. doi:10.1137/1.9781611973402.52.

[12] Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Trans. Algorithms*, 11(1):7:1–7:30, 2014. doi:10.1145/2636924.

[13] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. On interactivity in Arthur-Merlin communication and stream computation. *Electron. Colloquium Comput. Complex.*, 20:180, 2013. URL: http://eccc.hpi-web.de/report/2013/180.

[14] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable Stream Computation and Arthur-Merlin communication. In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 217–243. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CCC.2015.217.

[15] Amit Chakrabarti, Prantar Ghosh, and Justin Thaler. Streaming verification for graph problems: Optimal tradeoffs and nonlinear sketches. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.22.

[16] Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1–12. IEEE, 2020. doi:10.1109/FOCS46700.2020.00009.

[17] Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1327–1334. ACM, 2020. doi:10.1145/3357713.3384279.

[18] Lijie Chen and R. Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via PCPs of proximity. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 19:1–19:43. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.19.

[19] Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $AC^0 \circ mod_2$ lower bounds for the boolean inner product. *J. Comput. Syst. Sci.*, 97:45–59, 2018. doi:10.1016/j.jcss.2018.04.006.

[20] Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 47–58. ACM, 2016. doi:10.1145/2840728.2840734.

[21] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proc. VLDB Endow.*, 5(1):25–36, 2011. doi:10.14778/2047485.2047488.

[22] Yuval Filmus, Yuval Ishai, Avi Kaplan, and Guy Kindler. Limits of Preprocessing. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 17:1–17:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.17.

[23] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 260–270. IEEE Computer Society, 1981. doi:10.1109/SFCS.1981.35.

[24] Dmitry Gavinsky and Alexander A. Sherstov. A separation of NP and conp in multiparty communication complexity. *Theory Comput.*, 6(1):227–245, 2010. doi:10.4086/toc.2010.v006a010.

[25] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 86:1–86:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.86.

[26] Parikshit Gopalan, Adam R. Klivans, and David Zuckerman. List-decoding Reed-Muller codes over small fields. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 265–274. ACM, 2008. doi:10.1145/1374376.1374417.

[27] Tom Gur and Ran Raz. Arthur-Merlin streaming complexity. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 528–539. Springer, 2013. doi:10.1007/978-3-642-39206-1_45.

[28] Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 39:1–39:43. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.39.

[29] Johan Håstad. Almost optimal lower bounds for small depth circuits. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20. ACM, 1986. doi:10.1145/12130.12132.

[30] Xuangui Huang, Peter Ivanov, and Emanuele Viola. Affine extractors and AC0-Parity. *Electron. Colloquium Comput. Complex.*, page 137, 2021. URL: https://eccc.weizmann.ac.il/report/2021/137.

[31] Jeffrey C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.*, 55(3):414–440, 1997. doi:10.1006/jcss.1997.1533.

[32] Madhav Jha, C. Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 589–597. ACM, 2013. doi:10.1145/2487575.2487678.

[33] Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In Lusheng Wang, editor, *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, volume 3595 of *Lecture Notes in Computer Science*, pages 710–716. Springer, 2005. doi:10.1007/11533719_72.

[34] Stasys Jukna. On graph complexity. *Comb. Probab. Comput.*, 15(6):855–876, 2006. doi:10.1017/S0963548306007620.

[35] John Kallaugher, Andrew McGregor, Eric Price, and Sofya Vorotnikova. The complexity of counting cycles in the adjacency list streaming model. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 119–133. ACM, 2019. doi:10.1145/3294052.3319706.

[36] Daniel M. Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. Counting arbitrary subgraphs in data streams. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 598–609. Springer, 2012. doi:10.1007/978-3-642-31585-5_53.

[37] Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 118–134. IEEE Computer Society, 2003. doi:10.1109/CCC.2003.1214415.

[38] Hartmut Klauck. On arthur merlin games in communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 189–199. IEEE Computer Society, 2011. doi:10.1109/CCC.2011.33.

[39] Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Math.*, 8(1-2):161–185, 2012. doi:10.1080/15427951.2012.625260.

[40] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.

[41] Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better algorithms for counting triangles in data streams. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 401–411. ACM, 2016. doi:10.1145/2902251.2902283.

[42] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020. doi:10.1137/18M1195887.

[43] Ilan Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991. doi:10.1016/0020-0190(91)90157-D.

[44] Alexander A Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. *Math. notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[45] Guy N. Rothblum. How to compute under AC^0 leakage without secure hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, 2012. doi:10.1007/978-3-642-32009-5_32.

[46] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013. doi:10.1145/2488608.2488709.

[47] Ron D. Rothblum and Michael Ezra. Small Circuits Imply Efficient Arthur-Merlin Protocols. *Electron. Colloquium Comput. Complex.*, page 127, 2021. URL: https://eccc.weizmann.ac.il/report/2021/127.

[48] Rocco A. Servedio and Emanuele Viola. On a special case of rigidity. *Electron. Colloquium Comput. Complex.*, 19:144, 2012. URL: http://eccc.hpi-web.de/report/2012/144.

[49] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. doi:10.1145/1059513.1059516.

[50] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Comput. Complex.*, 15(4):298–341, 2006. doi:10.1007/s00037-007-0218-9.

[51] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

[52] Justin Thaler. Semi-streaming algorithms for annotated graph streams. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 59:1–59:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.59.

[53] Nikhil Vyas and R. Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of #SAT algorithms. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.59.

[54] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.

[55] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213. ACM, 1979. doi:10.1145/800135.804414.