On the Power of Regular and Permutation Branching Programs

Chin Ho Lee
Harvard University, Cambridge, MA, USA

Edward Pyne
MIT, Cambridge, MA, USA

Salil Vadhan
Harvard University, Cambridge, MA, USA

Abstract

We give new upper and lower bounds on the power of several restricted classes of arbitrary-order read-once branching programs (ROBPs) and standard-order ROBPs (SOBPs) that have received significant attention in the literature on pseudorandomness for space-bounded computation.

- Regular SOBPs of length $n$ and width $\lfloor w(n+1)/2 \rfloor$ can exactly simulate general SOBPs of length $n$ and width $w$, and moreover an $n/2 - o(n)$ blow-up in width is necessary for such a simulation.
- Our result extends and simplifies prior average-case simulations (Reingold, Trevisan, and Vadhan (STOC 2006), Bogdanov, Hoza, Prakriya, and Pyne (CCC 2022)), in particular implying that weighted pseudorandom generators (Braverman, Cohen, and Garg (SICOMP 2020)) for regular SOBPs of width $\text{poly}(n)$ or larger automatically extend to general SOBPs. Furthermore, our simulation also extends to general (even read-many) oblivious branching programs.
- There exist natural functions computable by regular SOBPs of constant width that are average-case hard for permutation SOBPs of exponential width. Indeed, we show that Inner-Product mod 2 is average-case hard for arbitrary-order permutation ROBPs of exponential width.
- There exist functions computable by constant-width arbitrary-order permutation ROBPs that are worst-case hard for exponential-width SOBPs.
- Read-twice permutation branching programs of subexponential width can simulate polynomial-width arbitrary-order ROBPs.

1 Introduction

Read-once branching programs (ROBPs) have been extensively studied over the past four decades, motivated by the fact that these programs capture how small-space machines use random coins, and hence optimal and explicit pseudorandom generators for them would imply $\text{BPL} = \text{L}$, showing that every randomized logspace algorithm can be simulated deterministically with only a constant factor blow-up in space. Thus, there has been several decades of research on constructing pseudorandom generators for different variants of ROBPs.
In this paper, we study how those variants compare to each other in computational power, through new simulations and separations. To describe our results, we first define the models we are studying, starting from the most general model of read-many branching programs.

**Definition 1.** An (oblivious) branching program (BP) $B$ of length $m$ and width $w$ computes a function $B : \{0, 1\}^n \to \{0, 1\}$. On an input $x \in \{0, 1\}^n$, the branching program computes as follows. It has $m + 1$ layers $V_0, \ldots, V_m$, each with vertices labeled $\{1, \ldots, w\}$. It starts at a fixed start state $v_{st} \in V_0$. Then for each step $t = 1, \ldots, m$, it reads the next symbol $x_{i(t)}$ for some $i(t) \in [n]$, and updates its state according to a transition function $B_t : V_{t-1} \times \{0, 1\} \to V_t$ by taking $v_t = B_t[v_{t-1}, x_{i(t)}]$. For $v \in V_s$ and $u \in V_t$ for $t > s$, we write $B[v, y] = u$ if the program transitions to state $u$ starting from state $v$ upon reading $y = (x_{i(s+1)}, \ldots, x_{i(t)})$. Moreover, there is a set of accept states $V_{acc} \subseteq V_m$. For $x \in \{0, 1\}^n$, we define $B(x) = 1$ if and only if $B[v_{st}, x] \in V_{acc}$. That is, $B$ accepts the inputs $x$ that lead it from the start state $v_{st} \in V_0$ in the first layer to an accept state in the last layer $v_{acc} \in V_{acc} \subseteq V_m$. We write $B(v, x) = 1$ if the program transitions to an accept state from a state $v$ on input $x$. We call the function $i : [m] \to [n]$ the read order of $B$.

**Definition 2.** A read-$k$ branching program is a $B$ where the read order $i$ satisfies $|i^{-1}(j)| \leq k$ for every $j \in [n]$. For $k = 1$ we denote this a read-once branching program (ROBP).

**Definition 3.** A standard-order ROBP (SOBP) is an ROBP whose read order is the identity function (i.e. $i(t) = t$ for every $t \in [n]$).

Note that we have the inclusions

$$\text{SOBPs} \subseteq \text{ROBPs} \subseteq \text{BPs}.$$ 

To emphasize the distinction between the standard-order model and general ROBPs (which have $i(t) = \pi(t)$ for some permutation $\pi$), we denote the latter as arbitrary-order ROBPs.

**Remark 4.** Our choice of notation follows the recent surveys of Hatami and Hoza [26] and Hoza [28]. There have been several (inconsistent) choices of notation in prior papers. In particular, prior works have referred to standard order ROBPs as simply ROBPs, or “ordered BPs”. Other works have referred to ROBPs as “unordered ROBPs”.

In 1990, Nisan [38] constructed an explicit pseudorandom generator (PRG) for SOBPs with seed length $O(\log n \cdot \log(\text{nw}/\varepsilon))$ (c.f. the optimal $O(\log(\text{nw}/\varepsilon))$ achieved by the probabilistic method). Despite extensive effort, this result has not been improved when the width of the programs $w$ is at least 4 and at most $2^{n^{o(1)}}$. Motivated by this longstanding challenge, researchers have extensively studied restricted cases of the model, known as regular and permutation SOBPs:

**Definition 5.** A regular BP is a branching program where for all $t \in [m]$ and $i \in [w]$, there are exactly two distinct pairs $(j_1, b_1), (j_2, b_2)$ such that $B_t[j_1, b_1] = B_t[j_2, b_2] = i$. Equivalently, the graph of transitions from $V_{t-1}$ to $V_t$ is 2-in-regular.

**Definition 6.** A permutation BP is an branching program where for all $t \in [m]$ and $\sigma \in \{0, 1\}$, $B_t[\cdot, \sigma]$ is a permutation on $[w]$.

Note that we have the inclusions

$$\text{permutation BPs} \subseteq \text{regular BPs} \subseteq \text{BPs}$$

and the same inclusions hold when restricting BPs to ROBPs or SOBPs.
There has been extensive prior work studying pseudorandomness for regular \([31, 8, 16, 5, 34, 12]\) and permutation \([32, 44, 41, 10, 29, 39, 22]\) SOBPs and ROBPs over roughly the last decade.

For regular SOBPs, the PRG of Braverman, Rao, Raz, and Yehudayoff \([8]\) improves on Nisan’s (which has seed length \(O(\log^2 n)\) even for \(w = 4\) and \(\varepsilon = 1/3\)) in the regime where both \(w\) and \(1/\varepsilon\) are subpolynomial, i.e. \(n^{o(1)}\). The later work \([5]\) obtained better seed length than \(O(\log^2 n)\) when either \(w\) or \(1/\varepsilon\) was \(n^{o(1)}\) (whereas Braverman et al. required both parameters to be small relative to \(n\)), at the cost of obtaining only a hitting set generator (HSG), a weaker object than pseudorandom generators that is sufficient for most derandomization tasks. For permutation SOBPs, Pyne and Vadhan \([39]\) achieved seed length \(\tilde{O}(\log^{3/2} n)\) for an object known as a weighted PRG,\(^1\) in the \(w = n\) regime motivated by derandomizing logspace.

Despite this extensive prior work, and the status quo where the known pseudorandom objects for regular and permutation SOBPs are better than those known for generic SOBPs in many regimes, there was relatively little work investigating the relative power of these models. As an example, it is well known that SOBPs can be simulated by a one-way two-party communication protocol, and therefore any program of width less than \(2^{\Omega(n)}\) cannot compute the Inner-Product function \(IP_{2n}(x) := \sum_{i=1}^{n} x_i x_{n+i} \pmod 2\) on average. However, this result does not say anything about the relative power of general SOBPs versus regular and permutation SOBPs.

1.1 Our Results

We begin a systematic study of the relative power of SOBPs, regular SOBPs, and permutation SOBPs. We first survey the landscape of known results before stating our results.

1.1.1 General vs. Regular Programs

Perhaps the best known upper bound in this regard is the work of Reingold, Trevisan, and Vadhan \([42]\) and its recent extension of Bogdanov, Hoza, Pyne, and Prakriya \([5]\). They showed that regular SOBPs of width \(\poly(nw)\) and length \(\tilde{O}(n)\) can approximately simulate general SOBPs of width \(w\) and length \(n\). This implies a “transfer result”: in the width-\(\poly(n)\) regime, optimal PRGs or HSGs for regular ROBPs imply the equivalent objects for general ROBPs, and hence for logspace computation. However, these results have a few limitations. First, the simulation was average-case, and due to this did not imply a transfer result for weighted PRGs, a pseudorandom object that has seen extensive recent interest \([7, 11, 14, 39, 27]\). Moreover, both proofs are relatively involved.

We show that this upper bound can be improved and substantially simplified, and in fact, general and regular programs of the same length \(m\), regardless of being read-once or not, are equivalent up to a factor of \(m\) in the width.

▶ Theorem 7 (Informal statement of Theorem 18). Let \(B\) be an oblivious branching program of length \(m\) and width \(w \geq 4\). There exists a regular oblivious branching program \(R\) of length \(m\) and width \(mw/2\) such that \(R(x) = B(x)\) for all \(x\). Moreover, \(R\) has the same read order as \(B\).

\(^1\) A weighted PRG is a tuple of functions \((G, \rho) : \{0,1\}^n \rightarrow \{0,1\}^n \times \mathbb{R}\), where the weighted expectation \(\mathbb{E}_x[\rho(x) \cdot B(G(x))]\) is within \(\varepsilon\) of \(\mathbb{E}[B(U_n)]\) for all \(B\) in the class.
As a consequence, weighted PRGs for regular SOBPs with seed length matching those known for permutation SOBPs [39] would imply an improved derandomization of logspace:²

▶ **Corollary 8.** Suppose there is an explicit weighted PRG for regular SOBPs of length \( n \) and width \( w \) with seed length \( \tilde{O}(\log n \cdot (\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon)) \). Then \( \text{BPL} \subseteq \text{L}^{1/3+o(1)} \).

This follows as a corollary of Theorem 7 and the argument of Chattopadhyay and Liao [11] that the Saks–Zhou algorithm [43] can be instantiated with a weighted PRG.

As mentioned above, Theorem 7 holds even for non-read-once branching programs (as defined in Definition 1), in contrast to the prior results of [42, 5]. As a corollary, we derive that \( \text{L} \) can be computed by polynomial width regular branching programs:

▶ **Corollary 9.** Every language in \( \text{L} \) can be decided by a (read-many) regular branching program of length and width \( \text{poly}(n) \), on inputs of size \( n \).

In terms of separation results, some simple observations were known. The AND function, which can be shown to require width \( n \) for permutation (in fact, regular) BP's, has a trivial general BP of width 2. (See Observation 19 for a proof.) We extend this separation to larger widths. This complements our simulation result (Theorem 7) by showing that in the case of ROBPs, the loss of a factor of \( n/2 \) is tight up to an additive term of \( (w \log w)/2 \).

▶ **Proposition 10.** For every \( w = 2^t, n \in \mathbb{N} \), there is a function \( f : \{0,1\}^n \rightarrow \{0,1\} \) computable by a general SOBP of width \( w \) such that every regular SOBP computing \( f \) has width at least \( \frac{nw}{2} - w \log w \).

It is known that general SOBPs of constant width cannot be approximated by regular SOBPs of some \( \text{poly}(n) \) width in the “sandwiching notion” [3]. This can be derived by combining the results of [9, 8]. Brody and Verbin [9] showed that there is an instantiation of the Impagliazzo–Nisan–Wigderson PRG [31] that does not fool general SOBPs of width 3, and yet Braverman et al. [8] shows that this same PRG fools regular SOBPs of width \( n^c \) for some \( c > 0 \).

### 1.1.2 Regular vs. Permutation Programs

For the relationship between permutation and regular SOBPs, the situation was even less clear. As discussed in the previous section, despite extensive work on pseudorandomness for permutation and regular SOBPs, prior work has not proven separations between the two models. In fact, as far as we know, prior work did not exhibit any function computable by a regular program that was not computable by a permutation program of equal width.

We develop new lower bounds that separate these models to a near-maximal extent.

▶ **Theorem 11.** There is \( c > 0 \) and \( w_0 \in \mathbb{N} \) such that for every \( \varepsilon > 0 \) and \( n \) the following holds. There exists a function \( f : \{0,1\}^n \rightarrow \{0,1\} \) computable by a regular SOBP of width \( w_0 \) such that no permutation SOBP of width \( 2^{c n / \log(1/\varepsilon)} \) agrees with \( f \) on a \( 1/2 + \varepsilon \) fraction of the inputs. In particular, no permutation SOBP of width \( 2^{c \sqrt{n}} \) agrees with \( f \) on a \( 1/2 + 2^{-\sqrt{n}} \) fraction of inputs.

² A preprint circulated by the second author claimed this as a consequence of [5]. However, it does not follow from the argument in that work.
The hard function in Theorem 11 is the Inner-Product function with a specific variable-ordering. Our techniques for proving Theorem 11 are information-theoretic, and rely on showing that the entropy of the state over the \( n + 1 \) layers of a permutation ROBP must be non-decreasing.

Our next result shows that the Inner-Product function is in fact average-case hard for arbitrary-order permutation ROBPs of exponential width.

▶ **Theorem 12.** Every arbitrary-order permutation ROBP \( B \) that computes \( \text{IP}^{\oplus n}(x) := \sum_{i=1}^{n} x_{2i-1}x_{2i} \pmod{2} \) on more than a \( 3/4 + \varepsilon \) fraction of inputs has width at least \( 2^{4\epsilon^2 n} \). Moreover, \( \text{IP}^{\oplus n} \) can be computed by a regular SOBP of width 4.

We conjecture that Theorems 11 and 12 can be strengthened to give optimal average-case hardness, namely \( 1/2 + 2^{-n} \), but we have not been able to prove such a result.

▶ **Conjecture 13.** There exists a constant \( c > 0 \) such that the following holds. Every arbitrary-order permutation ROBP \( B \) that computes \( \text{IP}^{\oplus n}(x) := \sum_{i=1}^{n} x_{2i-1}x_{2i} \pmod{2} \) on more than a \( 1/2 + 2^{-cn} \) fraction of inputs has width at least \( 2^{cn} \).

### 1.1.3 Standard-Order vs. Arbitrary-Order Programs

In the past decade, researchers have turned their attention to constructing PRGs from SOBPs to the more general model of arbitrary-order ROBPs, as a way to generate new ideas to improve the state-of-the-art PRGs for SOBPs, and to develop PRGs for several natural subclasses of circuits that are not captured by SOBPs, as circuit classes are closed under permutation of the input coordinates. This line of research has received extensive interests [30, 41, 45, 25, 36, 21, 19], and in particular has resulted in near-optimal PRGs for several well-studied models of computation, including read-once formulas [6, 23, 13, 17, 19], constant-width arbitrary-order permutation ROBPs [41, 10, 34], and read-once \( \mathbb{F}_2 \)-polynomials [35, 36, 33, 18].

While Theorem 12 shows that there are regular SOBPs which cannot be approximated by arbitrary-order permutation ROBPs of exponential width, we show that the opposite direction is also true, by giving a function that is computable by an arbitrary-order permutation ROBP of constant width that requires exponential width for (even general) SOBPs.

▶ **Proposition 14 (Informal statement of Proposition 35).** For every \( n \), there exists a function \( f : \{0,1\}^n \to \{0,1\} \) such that \( f \) is computable by an arbitrary-order permutation ROBP of width 6, and every SOBP computing \( f \) has width at least \( 2^{n/2} \).

This result uses a non-Abelian group product and an adversarial argument.

### 1.1.4 Read Once vs. Read Many

Given our exponential lower bounds (Theorems 11 and 12) for permutation ROBPs, it is natural to ask whether any of them extends to read-\( k \) programs.

We show that even in the read-2 setting, permutation branching programs already become substantially more powerful. Specifically, read-twice permutation branching programs of subexponential width can simulate arbitrary-order ROBPs of polynomial width:

▶ **Proposition 15.** Let \( f : \{0,1\}^n \to [w] \) be computable by an arbitrary-order ROBP \( B \) of width \( w \). Then for every \( k \in \mathbb{N} \), \( f \) is computable by a read-(2\(^k\)) permutation branching program \( B' \) of width \( w^{(k+1)n^{1/k}} \).
Our simulation in Proposition 15 follows directly from Bennett’s work on reversible computation [4]. We complement Proposition 15 by showing that a subexponential blow-up in the width is necessary for read-twice programs: there is no fixed read order such that read-twice permutation BPs reading bits in that order can simulate even regular SOBPs of constant width.

**Theorem 16.** For every read-twice ordering \( i : [2n] \to [n] \), there exists a function \( g : \{0, 1\}^n \to \{0, 1\} \) computable by a regular ROBP of width \( O(1) \), such that every read-twice permutation branching program \( P \) of width \( 2^{n^{1/8}} \) with read order \( i \) computes \( g \) correctly on at most \( 1/2 + 2^{-\Omega(n^{1/8})} \) fraction of inputs.

### 1.1.5 Permutation vs. Monotone Programs

Several works [37, 19] have studied the model of monotone branching programs, which correspond to branching programs where the edges labeled 1 do not cross, and likewise for the edges labeled 0. They are considered to be the “extreme opposite” of permutation programs [19]. We provide evidence for this belief by showing that read-once DNFs, which are computable by constant-width monotone programs, are worst-case hard for permutation SOBPs of exponential width:

**Proposition 17.** Let \( f(x_1, y_1, \ldots, x_n, y_n) = \bigvee_i (x_i \land y_i) \). Then every permutation SOBP computing \( f \) has width at least \( 2^n \).

## 2 Regular Branching Programs

We show that regular programs can exactly simulate general programs with a moderate blow-up in width. We emphasize that our simulation is not restricted to the read-once setting.

**Theorem 18.** Let \( B : \{0, 1\}^n \to \{0, 1\} \) be a branching program of length \( m \) and width \( w \). There is a regular branching program \( R : \{0, 1\}^n \to \{0, 1\} \) of length \( m \) and width \( w' := \max\{w, \frac{w m}{2} + w(1 - \log \frac{w}{2})\} \) such that \( R(x) = B(x) \) for all \( x \in \{0, 1\}^n \). Moreover, \( R \) has the same variable read order as \( B \). In particular, for \( w \geq 4 \), we have \( w' \leq \frac{w m}{2} \).

**Proof.** We prove by induction on length \( m \). We show the stronger claim that \( R \) exactly computes the states of \( B \), i.e., that there are maps \( \phi_t : [w'] \to [w] \) such that \( \phi_t(R[vst, x_{i(1)}, \ldots, x_{i(t)}]) = B[vst, x_{i(1)}, \ldots, x_{i(t)}] \) for every \( x \in \{0, 1\}^n \) and \( t \in [m] \).

When \( m \leq \log w \), we can simulate \( B \) trivially by storing the bits read in at most \( 2^n \leq w \) states. Now, suppose \( m \geq \log w + 1 \). For each state \( v \) in the \( (m - 1) \)-th layer \( B_{m-1} \) of \( B \), let \( C_{m-1}(v) := \phi_{m-1}^{-1}(v) \). By the inductive assumption, we have \( \sum_{v \in B_{m-1}} |C_{m-1}(v)| \leq w(m - 1)/2 + w(1 - \log(w)/2) \).

Now, for each state \( u \) in the \( m \)-th layer \( B_m \) of \( B \), create

\[
\left\lfloor \frac{1}{2} \sum_{(v, b) : B[v, b] = u} |C_{m-1}(v)| \right\rfloor
\]

states, denoted \( C_m(u) \), and define \( \phi_m \) such that \( \phi_m(C_m(u)) := u \).

Finally, for each \( b \in \{0, 1\} \) and \( v \in B_{m-1} \) such that \( B[v, b] = u \), we add a \( b \)-edge from every state in \( C_{m-1}(v) \) to some state in \( C_m(u) \). There are \( s_u := \sum_{(v, b) : B[v, b] = u} |C_{m-1}(v)| \) many such edges, and hence there are enough states in \( C_m(u) \) to accommodate this (with each state having at most 2 edges). Now, summing over all \( u \in B_m \), we have...
\[ |R_m| = \sum_{u \in B_m} |C_m(u)| \]
\[ = \sum_{u \in B_m} \left[ \frac{1}{2} \sum_{(v,b) \in E[v,b]=u} |C_{m-1}(v)| \right] \]
\[ \leq \sum_{u \in B_m} \left( \frac{1}{2} + \frac{1}{2} \cdot \sum_{(v,b) \in E[v,b]=u} |C_{m-1}(v)| \right) \]
\[ = \frac{|B_m|}{2} + \sum_{v \in V_{m-1}} |C_{m-1}(v)| \]
\[ \leq \frac{w(m-1)}{2} + w \left( 1 - \frac{\log w}{2} \right) + \frac{w}{2} \]
\[ = \frac{wm}{2} + w \left( 1 - \frac{\log w}{2} \right). \]

Let \( k \leq w \) be the number of \( u \) such that \( s_u \) is odd. Note that \( k \) must be even. For each such \( u \) there is a state in \( C_m(u) \) such that it has in-degree one. To preserve regularity, we add \( k/2 \leq w/2 \) of dummy states in \( R_{m-1} \) that are not reachable from the start state and connect the \( k \) outgoing edges of these states to these \( u \)’s. ▶

We now show that for general SOBPs, this loss of a factor of \( m \) is tight, and in fact the loss is even tight in the leading constant.

▶ **Proposition 10.** For every \( w = 2^t, n \in \mathbb{N} \), there is a function \( f : \{0,1\}^n \to \{0,1\} \) computable by an general SOBP of width \( w \) such that every regular SOBP computing \( f \) has width at least \( \frac{nm}{2} - w \log w \).

We recall the well-known fact that \( \text{AND}_n \) can be computed by a constant-width SOBP, but requires width \( n \) for regular ROBPs. We provide a proof for completeness.

▶ **Observation 19.** Given \( n \in \mathbb{N} \), \( \text{AND} := \text{AND}_n \) can be computed by a general SOBP of width 2. However, every regular SOBP \( R \) computing \( \text{AND} \) must have \( i + 1 \) distinct states reachable from \( v_{st} \) in layer \( i \).

**Proof.** The fact that \( \text{AND} \) can be computed by a general SOBP of width 2 is direct. We show the lower bound by induction. It is clearly true for layer 0 as \( v_{st} \) can reach itself. Assuming it holds for layer \( i \), we note that from correctness, \( u := R[v_{st}, 1^{i+1}] \neq R[v_{st}, 1^i][0] \) and hence there are two distinct states reachable in layer \( i + 1 \) from \( R[v_{st}, 1^i] \). Let \( R_i \) be the reachable states in layer \( i \) that are not \( R[v_{st}, 1^i] \). We have that there are at least \( 2|R_i| \) edges from \( R_i \) (and every endpoint of such an edge is reachable). Moreover, we claim that these edges cannot reach \( u \). Otherwise there would be \( \tau \neq 1^{i+1} \) such that \( B[v_{st}, \tau][1^{n-i-1}] = B[v_{st}, 1^n] \) which contradicts \( R \) computing \( \text{AND} \). Thus there are at least \( |R_i| + 1 \) vertices reachable in layer \( i + 1 \) that are not \( u \), so we conclude. ▶

We can then bootstrap this separation to work for larger widths. Essentially, we use a multiplexer to force the program to remember a large amount of information before computing \( \text{AND} \).

▶ **Definition 20.** Given \( n, w = 2^t \), let \( m = n - 2(t-1) \). Define \( f : \{0,1\}^{t-1} \times \{0,1\}^m \times \{0,1\}^{t-1} \to \{0,1\} \) as \( f(x, y, z) = (x, z) \oplus \text{AND}(y) = \sum_{i=1}^{t-1} x_i z_i + \text{AND}(y) \pmod{2} \).

We first argue that \( f \) can be computed by a SOBP of width \( w \).
On the Power of Regular and Permutation Branching Programs

> Claim 21. $f$ can be computed by an SOBP of width $w$.

Proof. We define a program $B(x, y, z)$. In the first $t$ layers, $B$ stores the entire input. For each state in layer $t$, $B$ uses 2 states to compute AND($y$), and hence at layer $m + t - 1$ the states are labeled $(x, \text{AND}(y))$. Then the program reads in $z$ and computes $(x, z)$, such that the states in the final layer are labeled $(x, z, \text{AND}(y))$ and hence $B$ can return the value of $f$. It is clear from this description that $B$ has width $2 \cdot 2^{t-1} = w$. <

We then argue that no regular SOBP can do better than remembering the first $t - 1$ bits, and moreover must compute AND using essentially disjoint states.

> Claim 22. For every regular SOBP $B$ computing $f$, for every $x \neq x' \in \{0, 1\}^{t-1}$ we have $B[v_{st}, x] \neq B[v_{st}, x']$. Furthermore, for every $k < m$ the states reachable in layer $t - 1 + k$ from $B[v_{st}, x]$ must be disjoint from those reachable from $B[v_{st}, x']$.

Proof. First assume for contradiction there are $x, x' \in \{0, 1\}^{t-1}$ with $x' \neq x$ where $B[v_{st}, x] = B[v_{st}, x']$. Let $i$ be some index where $x_i' \neq x_i$ and hence $(x, e_i) \neq (x', e_i)$. Thus, $f(x, 0^m, e_i) \neq f(x', 0^m, e_i)$, but

$$B[v_{st}, x][0^m || e_i] = B[v_{st}, x'][0^m || e_i]$$

which is a contradiction. For the second claim, assume for contradiction there are $\tau, \tau' \in \{0, 1\}^k$ (where we do not require $\tau \neq \tau'$) such that $B[v_{st}, x][\tau] = B[v_{st}, x'][\tau']$. Then $f(x, \tau || 0^{m-k}, e_i) \neq f(x', \tau' || 0^{m-k}, e_i)$ from before, but

$$B[v_{st}, x][\tau || 0^{m-k} || e_i] = B[v_{st}, x'][\tau' || 0^{m-k} || e_i]$$

which is a contradiction.<

We can then prove the result.

Proof of Proposition 10. Let $f$ be the function in Definition 20 with $n, w = 2^t$. By Claim 21, $f$ can be computed by a general SOBP of width $w$.

Now let $R$ be an arbitrary regular SOBP computing $f$. By Claim 22, we must have $R[v_{st}, x] \neq R[v_{st}, x']$ for every $x \neq x' \in \{0, 1\}^{t-1}$. Since $R$ must correctly compute AND($y$) (which can be shown by a similar extension argument), we obtain that for every $x$, there are at least $m$ states reachable from $R[v_{st}, x]$ in layer $t + m - 1$ for every $x$, and all of these states are disjoint by Claim 22. Thus, it follows from $m = n - 2(t - 1)$ that $R$ has width at least

$$2^{t-1} \cdot m = \frac{w}{2} \cdot m = \frac{nw}{2} + w(1 - \log w).$$<

3 Permutation Read-Once Branching Programs

In this section, we give explicit functions computable by small width regular SOBPs that are average-case hard against permutation SOBPs and ROBPs of large widths. We will be working with the Inner-Product functions with their input bits ordered in a certain manner.

Definition 23. For integers $\ell, m$, define $\text{IP}_{2^\ell}^m : \{0, 1\}^{2\ell} \to \{0, 1\}$ to be

$$\text{IP}_{2^\ell}^m (x^1, y^1, \ldots, x^m, y^m) := \bigoplus_{i=1}^m \langle x^i, y^i \rangle,$$

where $\langle x_1, \ldots, x_\ell, y_1, \ldots, y_\ell \rangle := \bigoplus_{j=1}^\ell x_j y_j$. We omit the subscript $2\ell$ when $\ell = 1$. 

We first show that $IP_{2^k}$ can be computed by a regular SOBP of width $2^{2k+1}$ via a simple argument. This follows from the fact that regular SOBPs can compute the XOR of an arbitrary function on $2\ell$ bits using $2\ell + 1$ bits, because we can store all the $2\ell$ bits and maintaining the prefix-XOR with 1 extra bit using a regular program. The program we construct is essentially the one used in simulating high-degree regular programs by binary regular programs in [5]:

► Lemma 24. Let $f : \{0,1\}^k \rightarrow \{0,1\}$ be an arbitrary function. Then $g : (\{0,1\}^k)^n \rightarrow \{0,1\}$ defined as
\[
g(x^1, \ldots, x^n) := \bigoplus_{i \in [n]} f(x^i),
\]
where $x^i \in \{0,1\}^k$ for each $i \in [n]$, can be computed by a regular SOBP of width $2^{k+1}$.

**Proof.** Let $B$ be a program where each state has label $(s, b) \in \{0,1\}^k \times \{0,1\}$. On reading $x_j^i$ where $j \in [k]$, the program updates as
\[
(s, b) \rightarrow \begin{cases} 
(s', b) & \text{if } 1 \leq j \leq k - 1 \\
(s', b \oplus f(s)) & \text{if } j = k.
\end{cases}
\]
where $s'$ is $s$ with the $j$-th coordinate replaced with the bit $x_j^i$. The width of this program is $2^k \cdot 2$, and the fact that it computes $f$ is direct. Finally, the program is regular as every $s'$ has a single $b \in \{0,1\}$ and two strings $s \in \{0,1\}^k$ for which the replacement of the $j$-th coordinate of $s$ with $b$ produces $s'$. ▶

We recall our average-case lower bound against permutation ROBPs computing inner product.

► Theorem 12. Every arbitrary-order permutation ROBP $B$ that computes \( IP_{n} \) defined as
\[
\sum_{i=1}^{n} x_{2i-1}x_{2i} \pmod{2}
\]
on more than a \( 3/4 + \epsilon \) fraction of inputs has width at least \( 2^{4\sqrt{n}} \). Moreover, \( IP_{n} \) can be computed by a regular SOBP of width 4.

For permutation SOBPs, we can strengthen this to a strong average case lower bound:

► Theorem 11. There is $c > 0$ and $w_0 \in \mathbb{N}$ such that for every $\epsilon > 0$ and $n$ the following holds. There exists a function $f : \{0,1\}^n \rightarrow \{0,1\}$ computable by a regular SOBP of width $w_0$ such that no permutation SOBP of width $2^{cn/\log(1/\epsilon)}$ agrees with $f$ on a $1/2 + \epsilon$ fraction of the inputs. In particular, no permutation SOBP of width $2^{c\sqrt{n}}$ agrees with $f$ on a $1/2 + 2^{-c/2}$ fraction of inputs.

Our argument relies on the fact that the entropy of the states in each layer of a permutation ROBP is non-decreasing. Before stating this property formally, we first recall some basic facts in information theory. We use capital letters to denote random variables, and lower case to denote specific assignments.

► Definition 25. Given a joint random variable $(X, Y)$, let
- $H(X) := \sum_{x \in \text{Supp}(X)} p(x) \log_2(1/p(x))$ be the (binary) entropy of $X$;
- $H(X \mid Y) := H(X, Y) - H(Y)$ be the conditional entropy of $X$ given $Y$, and
- $I(X; Y) := H(X) - H(X \mid Y)$ be the mutual information of $X$ and $Y$.
Moreover, given $p \in [0, 1]$, let $H(p) := p \log_2(1/p) + (1 - p) \log_2(1/(1 - p))$ be the entropy of a $p$-biased Bernoulli random variable.
We define the distributions over states of a program.

**Definition 26.** Given a ROBP $B$ of length $n$, for $i \in \{0, \ldots, n\}$, let $S_i$ be the distribution over the reachable states after reading $X_i$ of a uniformly random $X \sim \{0, 1\}^n$.

We then note the most important property of permutation SOBPs from this perspective: given the state reached after reading $x_i$ and the value of $x_i$, one can exactly recover the state after reading $x_{i-1}$. More generally, we have the following proposition.

**Proposition 27.** Let $(X_1, \ldots, X_n) \leftarrow U_n$. For every SOBP $B$ and $i < j$, we have $H(S_j \mid S_i, X_{i+1}, \ldots, X_j) = 0$. Moreover, if $B$ is a permutation SOBP then $H(S_i \mid S_j, X_{i+1}, \ldots, X_j) = 0$.

**Proof.** The first claim is immediate from the fact that knowing the current state $S_i$ and next $j - i$ bits $X_{i+1}, \ldots, X_j$ to be read determines the state $S_j$. The second claim is likewise immediate, as for a permutation SOBP there is exactly one state $S_i$ in layer $i$ that reaches the state $S_j$ in layer $j$ after reading $X_{i+1}, \ldots, X_j$.

We use this property to show that for permutation SOBPs, the entropy of the state at layer $i$ must increase by at least the mutual information between the state and the $i$-th input bit, and use this to conclude a lower bound on the width.

**Lemma 28.** Let $(X_1, \ldots, X_n) \sim \{0, 1\}^n$ be a uniform $n$-bit input. For a permutation SOBP $B$ of length $n$ and width $w$, let $i_1 < \cdots < i_m$ be some $m$ layers in $B$, and $X^{i_j} := (X_{i_j-1+1}, \ldots, X_{i_j})$, where $i_0 := 0$. Then

$$\log w \geq \sum_{j=1}^{m} I(X^{i_j} ; S_{i_j}).$$

**Proof.** We first prove that for every $j \in [m],$

$$H(S_{i_j}) = I(X^{i_j} ; S_{i_j}) + H(S_{i_j-1}).$$

Given this, the lemma follows from $H(S_0) = 0$ and

$$\log w = \log \text{supp}(S_m) \geq H(S_m) = \sum_{j=1}^{m} H(S_{i_j}) - H(S_{i_j-1}) = \sum_{j=1}^{m} I(X^{i_j} ; S_{i_j}).$$

We now prove Equation (1). By Proposition 27 we have

$$H(S_{i_j} \mid S_{i_j-1}, X^{i_j}) = 0 = H(S_{i_j-1} \mid S_{i_j-1}, X^{i_j}).$$

Applying the chain rule to both sides we obtain

$$H(S_{i_j-1}, X^{i_j}) = H(S_{i_j}, S_{i_j-1}, X^{i_j}) - H(S_{i_j} \mid S_{i_j-1}, X^{i_j})$$

$$= H(S_{i_j-1}, S_{i_j}, X^{i_j}) - H(S_{i_j-1} \mid S_{i_j}, X^{i_j})$$

$$= H(S_{i_j}, X^{i_j}).$$

Another chain rule to both sides gives

$$H(X^{i_j} \mid S_{i_j}) + H(S_{i_j}) = H(X^{i_j} \mid S_{i_j-1}) + H(S_{i_j-1}).$$
Thus,

\[
H(S_{i_j}) = H(X^{i_j} \mid S_{i_{j-1}}) + H(S_{i_{j-1}}) - H(X^{i_j} \mid S_{i_{j-1}})
\]

\[
= H(S_{i_{j-1}}) + (H(X_{i_j}) - H(X^{i_j} \mid S_{i_j})) - (H(X^{i_j}) - H(X^{i_j} \mid S_{i_{j-1}}))
\]

\[
= H(S_{i_{j-1}}) + I(S_{i_j}; X^{i_j}) - I(S_{i_{j-1}}; X^{i_j})
\]

\[
= H(S_{i_{j-1}}) + I(S_{i_j}; X^{i_j}),
\]

where the final step follows from the fact that \(X_{i_j}\) is independent of all prior bits, and thus the state at layer \(i_j\).

We are now prepared to prove the lower bounds. In both cases, we require Fano’s inequality. For the inner product bound, we use a simple formulation due to Regev [40]:

\[\text{Lemma 29 (Claim 2.1 [40]). Let } X \text{ be uniformly distributed over } \{0,1\}. \text{ Let } S \text{ be a random variable such that there exists } f \text{ such that } \Pr_{X,S}[f(S) \neq X] =: p \leq 1/2. \text{ Then } I(X;S) \geq 1 - H(p).\]

### 3.1 Mild Average-Case Lower Bounds for Arbitrary-Order Programs

We now prove the lower bound in Theorem 12: To illustrate the idea, consider a permutation SOBP \(B\) that reads its input \(x\) in the order of \(x_1, \ldots, x_{2n}\). We will show that when \(X\) is uniform over \(\{0,1\}^{2n}\), for every \(i \in [n]\), given the state \(S_{2i-1}\) reached by \(B\) after reading \(X_1, \ldots, X_{2i-1}\), we can use \(B\) to predict the value of \(X_{2i-1}\) better than random guessing, showing that there is non-trivial amount of mutual information between \(S_{2i-1}\) and \(X_{2i-1}\).

To see this, note that for every \(x \in \{0,1\}^n\),

\[
x_{2i-1} = |P^{\oplus 2n}(x_1, \ldots, x_{2i-1}, 0, x_{2i+1}, \ldots, x_{2n}) \oplus |P^{\oplus 2n}(x_1, \ldots, x_{2i-1}, 1, x_{2i+1}, \ldots, x_{2n}).
\]

Moreover, given a state \(S_{i+1}\), we can simulate the remaining program on the two inputs \((x_{2i} = 0, X_{2i+1}, \ldots, X_{2n})\) and \((x_{2i} = 1, X_{2i+1}, \ldots, X_{2n})\), for a uniform \(X_{2i+1}, \ldots, X_{2n}\), to compute the right hand side, which by a union bound, is correct and thus equals \(X_{2i-1}\) with probability at least \(1/2 + 2\varepsilon\).

**Proof of Theorem 12.** Let \(X = (X_1, \ldots, X_{2n}) := (X_1, Y_1, \ldots, X_n, Y_n)\) be a uniform random input. Fix an arbitrary-order permutation ROBP \(B\) that reads \(x\) in the order of \(x_{\sigma(1)}, \ldots, x_{\sigma(2n)}\) for some permutation \(\sigma\). By assumption we have \(\Pr[B(X) \neq f(X)] \leq 1/4 - \varepsilon\).

For every \(i \in [n]\), let \(r_i := \min\{\sigma^{-1}(2i - 1), \sigma^{-1}(2i)\}\) be the layer reached by \(B\) after reading the first bit of \(x_{2i-1}\) and \(x_{2i}\). Let \(L \subset [2n]\) be the indices of the variables of \(X\) read up to this point (i.e., \(L = \{\sigma(1), \ldots, \sigma(r_i)\}\)) and let \(R := [2n] \setminus L = \{\sigma(r_i + 1), \ldots, \sigma(2n)\}\).

We now show that \(I(X_{r_i}; S_{r_i}) \geq 1 - H(1/2 - 2\varepsilon)\), which suffices to prove the result by Lemma 28. To do so, given the state \(s_{r_i}\) in layer \(r_i\), we let our guess of \(x_{r_i}\) be

\[
g(s_{r_i}) = B[v, y^n] \oplus B[v, y^1],
\]

where \(y \leftarrow U_R\) is a random suffix and \(y^b\) is \(y\) with its \((t_b := \max\{\sigma^{-1}(2i - 1), \sigma^{-1}(2i)\})-th\) bit replaced with \(b \in \{0,1\}\). Observe that \(B[S_{r_i}, Y^{b+}]\) is identical to \(f(X)\) conditioned on \(X_{t_b} = b\). We have
On the Power of Regular and Permutation Branching Programs

\[ \Pr_X[g(S_{r_i}) \neq X_{r_i}] = \Pr_X[B[S_{r_i}, Y^{0}] \oplus B[S_{r_i}, Y^{1\ast}] \neq X_{r_i}] \]
\[ \leq \Pr_X[B[S_{r_i}, Y^{0}] \neq f(X)] + \Pr_X[B[S_{r_i}, Y^{1\ast}] \neq f(X)] \]
\[ \leq \Pr_X[B(X) \neq f(X) \mid X_{t_i} = 0] + \Pr_X[B(X) \neq f(X) \mid X_{t_i} = 1] \]
\[ = 2 \Pr_X[B(X) \neq f(X)] \]
\[ \leq 1/2 - 2\varepsilon. \]

By Lemma 29 we have \( I(X_{r_i}; S_{r_i}) \geq 1 - H(1/2 - 2\varepsilon) \geq 4\varepsilon^2 \). Therefore by Lemma 28 we have

\[ \log w \geq \sum_{i=1}^{n} I(X_{r_i}; S_{r_i}) \geq 4\varepsilon^2 n, \]

and hence \( w \geq 2^{4\varepsilon^2 n} \). The “moreover” claim follows from Lemma 24.

3.2 Moderate Average-Case Lower Bounds

Before proving our strong average-case lower bound (Theorem 11), we have to extend Theorem 12 to improve the correlation bound from 3/4 + \( \varepsilon \) to 1/2 + \( \varepsilon_0 \) for an arbitrary constant \( \varepsilon_0 \).

\textbf{Theorem 30.} Let \( \ell \geq 8 \log(1/\varepsilon) \). If \( B \) is a permutation SOBP of width \( w \) and length \( 2\ell m \) that agrees with \( \IP^{2m}_{2\ell} \) on a 1/2 + \( \varepsilon \) fraction of inputs, then \( w \geq 2^{\varepsilon m\ell/4} \).

The high-level idea is to combine the idea in the previous subsection with Goldreich–Levin list-decoding. Instead of predicting 1 bit, we will divide the input into blocks and show that we can predict the whole block of \( X^i \) given the state \( S_i \) reached by \( B \) upon reading \( X^i \). To do so, we first show that with probability at least \( \varepsilon/2 \) over all-but-the-\( Y^i \)-part of the input, we have the following property: Given the state \( S_i \), we can predict \( (X^i, Y^i) \) for a random sample \( Y^i \sim \{0, 1\}^\ell \) correctly with probability \( 1/2 + \varepsilon \). Then by the Goldreich–Levin theorem, we can use this predictor to narrow \( X^i \) down to a list of size \( 1/\varepsilon^2 \), showing that there is a non-trivial amount of mutual information between \( X^i \) and \( S_i \).

Our argument only requires the following bound on the “list size,” which follows from Parseval’s identity.

\textbf{Claim 31.} For every Boolean function \( f : \{0, 1\}^\ell \rightarrow \{0, 1\} \), there are at most \( 1/\varepsilon^2 \) many \( a \in \{0, 1\}^\ell \) such that \( \Pr[f(U) = \langle a, x \rangle] \geq 1/2 + \varepsilon/2 \).

\textbf{Proof.} This is equivalent to \( \tilde{f}(a) \geq \varepsilon \), where \( \tilde{f}(a) := \mathbb{E}_x[f(x)(-1)^{(a,x)}] \). Let \( L \) be the number of such \( a \)'s. Then by Parseval’s identity, we have \( L\varepsilon^2 \leq \sum_{a \in \{0, 1\}^\ell} \tilde{f}(a)^2 = \mathbb{E}[f(x)^2] \leq 1 \). Rearranging gives \( L \leq 1/\varepsilon^2 \). \( \diamond \)

3.2.1 Proof of Theorem 30

\textbf{Proof.} Let \( (X^1, \ldots, Y^m) \) be a uniformly random input of \( \IP^{2m}_{2\ell} \). Let \( B \) be a width-\( w \) permutation SOBP that agrees with \( \IP^{2m}_{2\ell} \) with probability \( 1/2 + \varepsilon \). Our goal is to show that \( w \geq 2^{\varepsilon m\ell/4} \). For \( i \in [m] \), let \( S_i \) denote the state \( B \) reaches after reading \( X^i \). We will show that \( I(S_i; X^i) \geq \varepsilon \ell/4 \), from which the theorem follows from Lemma 28.
To proceed, fix an \( i \in [m] \). Given an input \((x^1, y^1, \ldots, x^m, y^m)\) of \( B \), let \( z \in \{ \{0,1\}^{\ell}\}^{2m-1} \) denote all but the \( y^i \)-th block of \( y \), that is, \( z = (x^1, \ldots, y^{i-1}, x^i, x^{i+1}, \ldots, y^m) \). We will use the shorthand \( B(z, y^i) \) to denote \( B(x^1, y^1, \ldots, x^m, y^m) \). Given \( z \in \{ \{0,1\}^{\ell}\}^{2m-1} \) and an auxiliary bit \( a \in \{0,1\} \), consider the function \( B_{z,a} : \{0,1\}^{\ell} \rightarrow \{0,1\} \) defined by

\[
B_{z,a}(y^i) := B(z, y^i) \oplus \bigoplus_{j>i} (x^j, y^j) \oplus a.
\]

(One should think of \( a \) as a guess of the bit \( \bigoplus_{j<i} (x^j, y^j) \).) Let \( s_i \) be the state reached by \( B \) upon reading the prefix \((x^1, y^1, \ldots, x^i) \in \{ \{0,1\}^{\ell}\}^{2i-1} \). Observe that we can compute \( B_{z,a}(y^i) \) by simulating \( B \) starting from state \( s_i \) on the remaining inputs \((y^i, x^{i+1}, \ldots, y^m)\) and then XORing its output with \( a \).

We claim that with probability at least \( \varepsilon/2 \) over \((Z, A) \sim (\{0,1\}^{\ell}\}^{2m-1} \times \{0,1\} \), we have

\[
\Pr_{Y^i \sim \{0,1\}^{\ell}_i} \left[ B_{Z,A}(Y^i) = (X^i, Y^i) \right] \geq 1/2 + \varepsilon/2.
\]  

To see this, note that \( A \) is a correct guess of the bit \( \bigoplus_{j<i} (x^j, y^j) \) with probability 1/2, i.e. \( \Pr_{Y^i \sim \{0,1\}^{\ell}_i} \left[ \bigoplus_{j<i} (x^j, y^j) = A \right] = 1/2 \). Conditioned on \( A \) being the correct guess, it follows by an averaging argument that with probability at least \( \varepsilon/2 \) over \( Z \sim (\{0,1\}^{\ell}\}^{2m-1} \) we have

\[
\Pr_{Y^i \sim \{0,1\}^{\ell}_i} \left[ B_{Z,A}(Y^i) = (X^i, Y^i) \right] = \Pr_{Y^i \sim \{0,1\}^{\ell}_i} \left[ B(Z, Y^i) = (X^i, Y^i) + \bigoplus_{j>i} (x^j, y^j) \right]
\]

\[
= \Pr_{Y^i \sim \{0,1\}^{\ell}_i} \left[ B(Z, Y^i) = 1\mathrm{IF}^{\oplus k}(Z, Y^i) \right] \geq 1/2 + \varepsilon/2.
\]

Let us call the pair \((z, a)\) good if it satisfies Equation (3). Note that for a good \((z, a)\), by Claim 31, there are at most \( 1/\varepsilon^2 \) many choices of \( r \in \{0,1\}^\ell \) such that

\[
\Pr_{Y^i \sim \{0,1\}^{\ell}_i} \left[ B_{Z,a}(Y^i) = (r, Y^i) \right] \geq 1/2 + \varepsilon/2,
\]

and \( x^i \) is one of them, and thus we have the following claim.

\textbf{Claim 32.} \( H(X^i \mid S_i, (Z, A) \text{ good}) \leq \log(1/\varepsilon^2) \).

We will use the following fact behind the proof of Fano’s inequality.

\textbf{Claim 33 (Fano’s inequality).} Let \( X, Y, G \) be three random variables such that \( H(G \mid X, Y) = 0 \). Then

\[
H(X \mid Y) = H(G \mid Y) + H(X \mid G, Y).
\]

For a uniform \((X^1, \ldots, Y^m) \sim (\{0,1\}^{\ell})^{2m} \), let \( G := G(Z, A) \) be the indicator random variable of whether \((Z, A) \) is good. Let \( Z_{>i} \) denote \((X^{i+1}, \ldots, Y^m) \). Since \( X^i \) is independent of \( Z_{>i} \) and \( A \),

\[
H(X^i \mid S_i) = H(X^i \mid S_i, Z_{>i}, A).
\]

Now, given \( S_i, Z_{>i}, A, \) and \( X^i \), we can compute \( B_{Z,A} \) and determine if \((Z, A) \) is good, and thus we have \( H(G \mid S_i, X^i, Z_{>i}, A) = 0 \). So by Claim 33,

\[
H(X^i \mid S_i, Z_{>i}, A) = H(G \mid S_i, Z_{>i}, A) + H(X^i \mid G, S_i, Z_{>i}, A).
\]
44:14 On the Power of Regular and Permutation Branching Programs

We can bound the first term \(H(G \mid S_i, Z_{>i}, A)\) by \(H(G)\). For the second term, we apply Claim 32 as follows:

\[
H(X^i \mid S_i, Z_{>i}, A, G) = \Pr[G] \cdot H(X^i \mid S_i, Z_{>i}, A, G = 1) + (1 - \Pr[G]) \cdot H(X^i \mid S_i, Z_{>i}, A, G = 0)
\]

\[
\leq \Pr[G] \cdot \log(1/\varepsilon^2) + (1 - \Pr[G]) \cdot H(X^i).
\]

Applying both bounds to the right hand side of Equation (4) gives

\[H(X^i \mid S_i, Z_{>i}, A) \leq H(G) + \Pr[G] \cdot \log(1/\varepsilon^2) + (1 - \Pr[G]) \cdot H(X^i).\]

As \(\Pr[G] \geq \varepsilon/2\), we have \(H(G) \leq 2 \Pr[G] \log(1/\Pr[G]) \leq 2 \Pr[G] \log(2/\varepsilon)\). Therefore,

\[I(X^i; S_i) = H(X^i) - H(X^i \mid S_i)
\]

\[= H(X^i) - H(X^i \mid S_i, Z_{>i}, A)
\]

\[\geq \Pr[G] \cdot (H(X^i) - \log(1/\varepsilon^2)) - H(G)
\]

\[\geq \Pr[G] \cdot (H(X^i) - 4 \log(1/\varepsilon))
\]

\[\geq (\varepsilon/2) \cdot (\ell - 4 \log(1/\varepsilon)),
\]

which is at least \(\varepsilon \cdot \ell/4\) for \(\ell \geq 8 \log(1/\varepsilon)\). It follows from Lemma 28 that

\[\log w \geq \sum_{i=1}^m I(S_i; X^i) \geq \varepsilon m \ell/4.\]

### 3.3 Strong Average-Case Lower Bounds

We now prove Theorem 11. Assadi and N. [1] proved the following XOR Lemma for multi-pass streaming algorithms.\(^3\) In the case of one pass this is essentially the same as SOBPs. Moreover, we observe that their argument also applies to permutation SOBPs.

**Lemma 34** ([1]). There exists an absolute constant \(\varepsilon_0 > 0\) such that the following holds. Let \(f: \{0,1\}^m \to \{0,1\}\) be any function and let \(f^\otimes \ell\) be the XOR of \(\ell\) copies of \(f\) on disjoint (sequential) blocks. Suppose \(\Pr[P(U) = f(U)] \leq 1/2 + \varepsilon\) for some \(\varepsilon \leq \varepsilon_0\) for every permutation SOBP \(P\) of width \(w\). Then \(\Pr[P(U) = f^\otimes \ell(U)] \leq 1/2 + \varepsilon/\ell\) for every permutation SOBP \(P\) of width \(w\).

**Proof of Theorem 11.** By Theorem 30, for every constant \(\varepsilon_0 > 0\), there exists a constant \(\ell\) such that the function \(\text{IP}_{2\ell}^{\oplus m}\) on \(2\ell m\) bits is \((1/2 + \varepsilon_0)\)-hard for permutation SOBPs of width \(2^{c_{m/\log(1/\varepsilon)}}\). By Lemma 34, the function \(\text{IP}_{2\ell}^{\oplus mk}\) on \(2\ell mk\) bits is \((1/2 + \varepsilon_0/k^{1/\ell})\)-hard for permutation SOBPs of width \(2^{c_{m/\log(1/\varepsilon)}}\). Choosing \(\varepsilon_0\) to be a sufficiently small constant, \(k = 7 \log_{\varepsilon_0} (1/\varepsilon)\), and letting \(n := 2\ell mk\) gives us a hard function on \(n\) bits that is \((1/2 + \varepsilon)\)-hard for permutation SOBPs of width \(2^{c_{m/\log(1/\varepsilon)}}\) for a universal constant \(c\).

### 3.4 Worst Case Lower Bounds Against Monotone Functions

Next, we show there are monotone functions (in fact, read-once DNFs) that are worst-case hard for permutation SOBPs of exponential width.

**Proposition 17.** Let \(f(x_1, y_1, \ldots, x_n, y_n) = \bigvee_i (x_i \land y_i)\). Then every permutation SOBP computing \(f\) has width at least \(2^n\).

\(^3\) There is a mistake in the publicly available versions which has been corrected by the authors [2].
Proof. Let $B$ be a permutation SOBP computing $f$. We will show that in layer $2i$ (the state after reading both variables in the $i$th term), there are $2^i$ states reachable by strings that have not yet satisfied a term. This holds vacuously for $i = 0$. Now suppose this holds for term $i$ and let $T_i$ be the set of such states, and for $b \in \{0, 1\}$ define

$$T_i[b] := \{v \in V_{2i+1} : \exists u \in T_i \text{ s.t. } B[u, b] = v\}.$$ 

We first observe that $|T_i[1]| = |T_i[0]| = |T_i| \geq 2^i$ and $T_i[1] \cap T_i[0] = \emptyset$. The first follows since $B$ is a permutation SOBP (and hence all states in $T_i[1]$ can have a single in-1-edge and likewise for $T_i[0]$) and the second follows via an extension argument, since otherwise $B$ fails to compute $f$. This implies $|T_i[10] \cup T_i[00]| \geq 2|T_i|$ again using that $B$ is a permutation SOBP. Finally, we observe that $T_{i+1} \supseteq T_i[10] \cup T_i[00]$ and hence $|T_{i+1}| \geq 2^{i+1}$ as claimed, which completes the induction.

The “moreover” claim follows from inspection. ▶

3.5 Separating General SOBPs From Permutation ROBPs

We now give a function $f$ that is computable by an arbitrary-order permutation ROBP of constant width but is hard for any SOBP of exponential width.

▶ Proposition 35. Let $D_3$ be the Dihedral group of order 6 with identity element $e$ and fix two reflections $r, s$ such that $r^2 = s^2 = e$ and $rs \neq sr$. Let $S = \{r, s, rs\}$. Consider $f: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ defined by

$$f(x, y) := \#(r^{x_1}s^{b_1} \cdots r^{x_n}s^{b_n} \in S).$$

Then every general SOBP computing $f$ has width at least $2^n$. Moreover, $f$ can be computed by an arbitrary-order permutation ROBP of width 6.

Proof. The “moreover” claim follows from the fact that any group product can be simulated by a permutation ROBP of width equal to the group’s order. We now claim that for every $x \neq x' \in \{0, 1\}^n$, there is a $y \in \{0, 1\}^n$ such that $f(x, y) \neq f(x', y)$, and therefore any SOBP must use $2^n$ states to remember $x$ after reading it.

First, consider the case where the Hamming weights of $x$ and $x'$ have different parities. Then by taking $y$ to be the all-zero string $0^n$ and using $r^2 = e$, we have $f(x, y) = r^b$ and $f(x', y) = r^{1-b}$ for some $b \in \{0, 1\}$.

Now, suppose their parities are the same. Let $i \in [n]$ be the first position where $x$ and $x'$ differ, and without loss of generality assume $x_i = 1$ (and so $x'_i = 0$). Let $y = e_i$. Then $f(x, y) = rsr^b$ and $f(x', y) = sr^{1-b}$ for some $b \in \{0, 1\}$, but $s, sr \in S$ and $rsr, rs \notin S$. So in either case we have $f(x, y) \neq f(x', y)$. ▶

References


On the Power of Regular and Permutation Branching Programs


We first show that read-twice permutation branching programs can compute polynomial width arbitrary-order ROBPs in subexponential width. This follows from Bennett’s simulation in reversible computation [4], but here we have to take into account the number of reads of the program. We remark that the proof of this result implicitly uses permutation branching programs where the bound on the number of accept states is much smaller than the width bound [29].

▶ Proposition 15. Let \( f : \{0,1\}^n \to [w] \) be computable by an arbitrary-order ROBP \( B \) of width \( w \). Then for every \( k \in \mathbb{N} \), \( f \) is computable by a read-(2^k) permutation branching program \( B' \) of width \( w^{(k+1)n^{1/k}} \).

Proof. As both programs can read inputs in arbitrary order, by permuting the indices of \( x \) we can without loss of generality assume \( B \) is a SOBP. Given a SOBP \( O \) of width \( w \), we first how to simulate it using a permutation read-2 BP \( P \) of width \( w^m \) for some \( m \leq \sqrt{2n} \), in which every state is represented by an \( m \)-tuple in \([w]^m\). We will show that every step in \( P \) is reversible, that is, for every state \( s_{t-1}^p \in [w]^m \) in \( P \), not only can we compute \( s_t^p := B_t(s_{t-1}^p, x_{i(t)}) \) given \( s_{t-1}^p \), but we can also compute \( s_{t-1}^p \) given \( s_t^p \) and \( x_{i(t)} \), where \( i(t) \in [n] \) is the coordinate of \( x \) read by \( P \) at the \( t \)-th step. One can verify that this is equivalent to the condition that \( P \) is a permutation program.
We construct $P$ as follows. At each step, $P$ remembers the at most $m$ out of the $n$ states reached by $O$. Let $s_i \in [w]$ be the state reached by $O$ after reading $x_1, \ldots, x_i$. The program $P$ first reads $x_1, \ldots, x_m$ to compute and store the $m$ states $(s_1, \ldots, s_m) \in [w]^m$ reached by $O$ in the first $m$ steps. Knowing $s_{m-2}$, we can compute $s_{m-1}$ again to erase $s_{m-1}$ from the memory, reaching the state $(s_1, \ldots, s_{m-2}, 0, s_m)$. Knowing $s_{m-3}$, we can compute $s_m$ again to erase $s_m$, reaching $(s_1, \ldots, s_{m-3}, 0, 0, s_m)$.

Now, given $s_m$, we read the next $m - 1$ bits $x_{m+1}, \ldots, x_{2m-1}$ to compute and store $(s_{m+1}, \ldots, s_{2m-1}, s_m)$. Using a similar strategy, we can read $x_{2m-2}, \ldots, x_m$ again to erase $s_{2m-2}, \ldots, s_{m+1}$ from memory, giving us $(0, \ldots, 0, s_{2m-1}, s_m)$.

Continuing, we can compute $(s \sum_{i=1}^m i, s \sum_{i=2}^m i, \ldots, s_{2m-1}, s_m)$ reversibly with $w^m$ states. Thus we can compute $s_n$ as long as

$$\sum_{i=1}^m i = \frac{m(m+1)}{2} \geq n.$$ 

which holds when $m = \sqrt{2n}$.

We just showed how to compute the $m$-tuple of states $(s \sum_{i=1}^m i, s \sum_{i=2}^m i, \ldots, s_{2m-1}, s_m)$ reversibly by reading the input twice. By reading the input another two times, from $(s \sum_{i=1}^m f(i), s \sum_{i=2}^m f(i), \ldots, s_{2m-1}, s_m)$ we can erase everything but $s \sum_{i=1}^m i =: s f(m)$ to compute $(s f(m), 0, \ldots, 0)$ reversibly. We now repeat the above strategy recursively to compute $(s \sum_{i=1}^m f(i), s \sum_{i=2}^m f(i), \ldots, s_{2f(m)-1}, s f(m))$ with a read-4 permutation program of width $\sum_{i=1}^m f(i)$.

By an inductive argument, we can compute the state $s_n$ of the read-once branching program reversibly with a read-(2$^k$) permutation program of width $w^m$, whenever

$$n \leq \sum_{i=1}^m i_1 \sum_{i_2=1}^{i_1} i_2 \cdots \sum_{i_k=1}^{i_{k-1}} i_k = \binom{m+k}{k+1}.$$ 

Choosing $m \geq (k+1)n^{1/(k+1)}$ completes the proof.

### A.1 Hardness for Read-Twice Permutation Programs

We now show that an exponential blow-up in the width in Proposition 15 is necessary. We first restate the theorem.

\begin{theorem}
For every read-twice ordering $i : [2n] \to [n]$, there exists a function $g : \{0, 1\}^n \to \{0, 1\}$ computable by a regular ROBP of width $O(1)$, such that every read-twice permutation branching program $P$ of width $2^{n/8}$ with read order $i$ computes $g$ correctly on at most $1/2 + 2^{-\Omega(n/8)}$ fraction of inputs.
\end{theorem}

\footnote{Note that the order of how the bits are read matters. For example, without knowing $s_{m-2}$ we cannot compute $s_{m-1}$ using $x_{m-1}$.}
A 2-pass BP is a read-2 BP where the first read of all its $n$-bit input come before the second read of any bit. We first prove an average-case lower bound against 2-pass permutation programs of width $2^{\sqrt{n}}$, where the second pass of the $n$-bit input is read in the same or the reverse order as the first pass. We show that given any read-twice ordering of the input bits, either $\sqrt{n}$ of the bits can be read in a read-once manner, or $n^{1/4}$ of the bits can be read in the 2-pass manner described above. In either case, we can define our hard function on at least $n^{1/4}$ bits and apply our average-case lower bounds. As both programs in the theorem can read input bits in arbitrary order, by permuting the indices of the input we can assume the indices in the first pass of the read are in increasing order.

### A.1.1 From 2-pass lower bound to read-once lower bound

We obtain our 2-pass lower bound by a reduction to our read-once lower bound (Theorem 11) based on an idea by David, Papakonstantinou, and Sidiropoulos [15].

**Proposition 36.** Let $P$ be a 2-pass permutation BP of width $w$ that reads its first pass of the input in the standard order, and its second pass in the same or reverse order as the first pass. If $\Pr[P(U) = f(U)] \geq 1/2 + \varepsilon$ for some $f : \{0, 1\}^n \rightarrow \{0, 1\}$, then there exists a permutation SOBP permutation program $P'$ of width $w^2$ such that $\Pr[P'(U) = f(U)] \geq 1/2 + \varepsilon/w$.

**Corollary 37.** There exists a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ computable by a regular SOBP of constant width that is $(1/2 + 2^{-\sqrt{n}})$-hard against 2-pass permutation BPs of width $2^{\sqrt{n}}$ that reads its second pass of the input in the same or reverse order as the first pass for a universal constant $c$.

**Proof.** Let $f$ be the function in Theorem 11 with $\varepsilon = 2^{-1/(1+c)\sqrt{n}}$, which is hard against permutation SOBPs of width $2^{(c/(1+c))\sqrt{n}}$ for some universal constant $c$. Suppose $f$ is not $(1/2 + 2^{-\sqrt{n}})$-hard against a 2-pass permutation SOBP of width $w = 2^{\sqrt{n}}$. Then by Proposition 36, $f$ is not $(1/2 + 2^{-1/(1+c)\sqrt{n}})$-hard against a permutation SOBP of width $2^{2\sqrt{n}}$. Choosing $c$ such that $2c < c/(1 + c)$, we get a contradiction.

**Proof of Proposition 36.** We first handle the case where the second pass is in the same order as the first pass. Suppose

$$\Pr[P(U) = f(U)] - \Pr[P(U) \neq f(U)] > \varepsilon.$$

Let $V_n$ be the layer $P$ reaches after making its first pass on $x$. By an averaging argument, there must be a state $v^* \in V_n$ such that

$$\Pr[(P(U) = f(U)) \land P_{\rightarrow v^*}(U)] - \Pr[(P(U) \neq f(U)) \land P_{\rightarrow v^*}(U)]$$

$$\geq \frac{1}{|V_n|} \sum_{v \in V_n} \left( \Pr[(P(U) = f(U)) \land P_{\rightarrow v}(U)] - \Pr[(P(U) \neq f(U)) \land P_{\rightarrow v}(U)] \right)$$

$$\geq \frac{1}{|V_n|} \left( \Pr[P(U) = f(U)] - \Pr[P(U) \neq f(U)] \right)$$

$$\geq \frac{\varepsilon}{|V_n|}.$$

For $b \in \{0, 1\}$, consider the new function $P^b_\varepsilon$ that outputs $P(x)$ if $P(x) = 1$ and outputs $b$ otherwise. Note that $\text{adv}(P^b_\varepsilon, f) \geq \varepsilon/|V_n|$ for one of the $b \in \{0, 1\}$. Assume $b = 0$ without loss of generality.
We now show that the function $P'_0$ can be computed by a permutation SOBP of width $w^2$ as follows. Its $i$-th layer $V'_i$ is $V_i \times V_{n+i}$. Its start state is $(v_0, \nu^*)$. Its accept states are $V'_{acc} := \{(v_1, v_2) : (v_1 = \nu^* \land v_2 \in V_{acc})\}$.

To handle the case where the second pass is in the reverse order, we use a similar idea. Suppose

$$\Pr[P(U) = f(U)] - \Pr[P(U) \neq f(U)] > \varepsilon.$$ 

Let $V_{acc} \subseteq V_{2n}$ be the set of accept states in the final layer. By an averaging argument, there must be a state $\nu^* \in V_{acc}$ such that

$$\Pr[P_{\nu^*}(U) = f(U)] - \Pr[P_{\nu^*}(U) \neq f(U)] \geq \frac{\varepsilon}{|V_{acc}|}.$$ 

We now show that the function $P_{\nu^*}$ can be computed by a permutation SOBP program of width $w^2$. Here we use the fact that $P$ is a permutation BP, where we can reverse the transitions in the program as follows. Define the reversed transition $P^{-1}_r : V_r \times \{0, 1\} \to V_{r-1}$ to be $P^{-1}_r[v_r, x_r] := v_{r-1}$, where $v_{r-1}$ is the unique state $v \in V_{r-1}$ such that $P_r[v_{r-1}, x_r] = v_r$.

To implement $P_{\nu^*}$, its $i$-th layer $V_i$ is $V_i \times V_{2n-i}$. Its start state is $(v_0, \nu^*) \in V_0 \times V_{2n}$. Its transition $P^r_0 : V_{i-1} \to V'_i$ is $P^r_0((v_1, v_2), x_i) = (P^r_1(v_1, x_i), P^{-1}_{n-i+1}(v_2, x_i))$. Its accept states are $V^r_{acc} := \{(v_1, v_2) : v_1, v_2 \in V_n : v_1 = v_2\}$. ▶

### A.1.2 From 2-pass lower bound to read-2 lower bound

We follow a similar idea that is used in [24]. Given a read-2 sequence, by permuting the indices of the input bits, we may assume the first pass is in increasing order. We will show that it contains a subsequence of the form $i_1 i_1 i_2 i_2 \cdots i_\sqrt{n} i_\sqrt{n}$, in which case we can define the hard function on $x_{i_1}, \ldots, x_{i_\sqrt{n}}$ and applying our read-once lower bound on $\sqrt{n}$ bits, or it contains a 2-pass subsequence of the form $i_1 \cdots i_\sqrt{n} i_{\sigma(1)} \cdots i_{\sigma(\sqrt{n})}$ for some permutation $\sigma : [\sqrt{n}] \to [\sqrt{n}]$, in which case by the Erdős–Szekeres theorem (Theorem 38 below), the sequence $i_{\sigma(1)} \cdots i_{\sigma(\sqrt{n})}$ must contain a monotone subsequence $i_{\mu(1)} \cdots i_{\mu(\sqrt{n})}$ of length $n^{1/4} - 1$, and so the read-2 sequence contains a 2-pass sequence on $n^{1/4} - 1$ bits where the second pass is in the same or reverse order as the first pass. So we can apply our 2-pass lower bound.

Before proving Theorem 16, we first state the Erdős–Szekeres theorem, which will be used in our proof.

**Theorem 38 (Erdős–Szekeres [20]).** For any integers $s$ and $r$, any sequence of distinct real numbers of length $sr + 1$ contains a monotonically increasing subsequence of length $s + 1$ or a monotonically decreasing subsequence of length $r + 1$.

**Proof of Theorem 16.** Let $s \in [n]^{2n}$ be a read-2 sequence. For $i \in [n]$, let $\text{pos}^1(i)$ and $\text{pos}^2(i)$ be the locations of the first and second occurrence of $i$, respectively. Partition the $2n$ indices of the sequence into $\sqrt{n}$ blocks $B_k : k \in [\sqrt{n}]$, where $B_k := [\text{pos}^1((k-1)\sqrt{n})+1) : \text{pos}^1(k\sqrt{n})-1]$. We consider two cases.

Suppose each block contains both occurrences of some element. That is, for each block $B_k : k \in [\sqrt{n}]$, we have $\text{pos}^1(i_k), \text{pos}^2(i_k) \in B_k$ for some $i_k \in [n]$. Then $s$ contains the subsequence

$$i_1 i_1 \cdots i_{\sqrt{n} i_{\sqrt{n}} \cdots i_{\sqrt{n}}}.$$
We define \( g(x) := f(x_{i_1}, \ldots, x_{i_{\sqrt{n}}}) \), where \( f \) is the hard function defined in Theorem 11 (but on \( \sqrt{n} \) bits). Let \( P \) be a read-2 permutation BP of width \( 2n^{1/8} \leq 2^{n^{1/4}} \) which reads its input in the order given by \( s \). By Theorem 11, we have that \( \Pr[P(U) = g(U)] \leq 1/2 + 2^{-\Omega(n^{1/4})} \) and \( g \) is computable by a regular ROBP of constant width.

Otherwise, some block does not contain both occurrences of any element. In other words, there exists a block \( B_k \) such that none of \( \text{pos}^2((k-1)\sqrt{n} + 1), \ldots, \text{pos}^2(k\sqrt{n}) \) lies in \( B_k \). In this case, \( s \) contains the 2-pass subsequence
\[
((k-1)\sqrt{n} + 1) \cdots (k\sqrt{n}) \cdot \sigma((k-1)\sqrt{n} + 1) \cdots \sigma(k\sqrt{n})
\]
for some permutation \( \sigma \) on the \( \sqrt{n} \) elements in the subsequence. Applying the Erdős–Szekeres theorem to the second half of the subsequence, we obtain a 2-pass subsequence on \( n^{1/4} - 1 \) elements from \( s \) where the second pass in the same or reverse order as the first pass. As in the previous case, by defining \( g \) to be the hard function \( f \) in Theorem 11 on these \( n^{1/4} - 1 \) bits, we conclude that \( \Pr[P(U) = g(U)] \leq 1/2 + 2^{-\Omega(n^{1/8})} \) for any read-twice permutation program reading its input in the order given by \( s \), and \( g \) is computable by a regular ROBP of constant width.