Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 97 (2023)

Efficient Interactive Proofs for Non-Deterministic Bounded Space

Joshua Cook* Ron D. Rothblum†

August 28, 2023

Abstract

The celebrated **IP** = **PSPACE** Theorem gives an efficient interactive proof for any bounded-space algorithm. In this work we study interactive proofs for *non-deterministic* bounded space computations. While Savitch's Theorem shows that nondeterministic bounded-space algorithms can be simulated by deterministic bounded-space algorithms, this simulation has a quadratic overhead. We give interactive protocols for nondeterministic algorithms directly to get faster verifiers.

More specifically, for any non-deterministic space S algorithm, we construct an interactive proof in which the verifier runs in time $\tilde{O}(n + S^2)$. This improves on the best previous bound of $\tilde{O}(n + S^3)$ and matches the result for *deterministic* space bounded algorithms, up to polylog(S) factors.

We further generalize to *alternating* bounded space algorithms. For any language L decided by a time T, space S algorithm that uses d alternations, we construct an interactive proof in which the verifier runs in time $\tilde{O}(n + S\log(T) + Sd)$ and the prover runs in time $2^{O(S)}$. For $d = O(\log(T))$, this matches the best known interactive proofs for deterministic algorithms, up to polylog(S) factors, and improves on the previous best verifier time for nondeterministic algorithms by a factor of $\log(T)$. We also improve the best prior verifier time for *unbounded* alternations by a factor of S.

Using known connections of bounded alternation algorithms to bounded depth circuits, we also obtain faster verifiers for bounded depth circuits with *unbounded* fan-in.

*University of Texas at Austin. Email: jac22855@utexas.edu.
†Technion. Email: rothblum@cs.technion.ac.il.

ISSN 1433-8092

Contents

1 Introduction

Interactive proofs, introduced by Goldwasser Micali and Rackoff [GMR89], are proof systems that enable a prover to convince a verifier of the truth of a given statement. The interaction proceeds in rounds where in each round the prover sends a message and the verifier responds. Crucially, in every round the verifier uses randomness that the prover cannot predict. At the end of the interaction the verifier either accepts or rejects the statement. We require that the honest prover convinces the verifier to accept true statements with high probability (and in fact, in most[1] protocols with probability 1) and that no prover, even a computationally unbounded one, can convince the verifier to accept a false statement other than with some small probability.

One of the most celebrated results in complexity theory is that **IP** = **PSPACE** [Lun+90; Sha92]. That is, the set of languages with polynomial space algorithms is exactly the set of languages with interactive protocols whose verifiers run in polynomial time. Interactive proofs have been prolific throughout other areas of complexity theory, including circuit lower bounds [San07; MW18], pseudorandomness from uniform assumptions [TV02], and has also been very influential in other proof systems, such as MIPs [BFL90], PCPs [Bab+91; Fei+91; AS98; Aro+98], and IOPs [BSCS16; RRR16].

The **IP** = **PSPACE** result can be generalized to any deterministic bounded space computation. For a space S deterministic algorithm, the interactive protocols with the fastest verifiers [Coo22b; Tha20] have a time $\tilde{O}(n + S^2)$ verifier and time $2^{O(S)}$ prover, where $\tilde{O}$ hides polylog(S) factors.[2]

In this work we study interactive proofs for more general forms of bounded space computations: *non-deterministic bounded space* and *alternating bounded space*. Recall that a non-deterministic space S algorithm is a space S Turing machine that gets in addition *read-once* access to a witness (which can be as long as 2^S). For example, the complexity class **NL** refers to non-deterministic logarithmic-space algorithms. Alternating algorithms are a generalization of nondeterministic algorithms that can 'alternate' quantifiers. The prior best protocols [Coo22b] for space S nondeterministic algorithms have verifier time $\Omega(n + S^3)$, which is an S factor slower than the best verifiers for deterministic algorithms. See Table 1 for a more complete comparison with prior works, and Section 2 for further details.

1.1 Our Results

Our main result is an improved interactive proof for alternating algorithms. We start by highlighting a special-case of this result for nondeterministic bounded-space algorithms. We construct interactive proofs for space S nondeterministic algorithms whose verifier runs in time $\tilde{O}(n + S^2)$, matching the time bound for deterministic verifiers (up to polylog(S) factors). Broadly, our techniques combine the recent verifier efficient interactive proofs for bounded space by Cook [Coo22b], with an efficient interactive proof for $\mathbf{AC}^0_\oplus$ circuits of Goldreich and Rothblum [GR20], and an improved derandomization through random walks on expander graphs.

The new interactive proof for non-deterministic bounded space is a special case of a more general result that we show for alternating bounded space algorithms. To state the result precisely, we first set up the notation. Let $\mathbf{ATISP}_d[T, S]$ be the set of languages decided by a simultaneous time T, space S and d alternation algorithm. Alternating algorithms have 3 tapes, a read only input tape containing the input, a read once input containing a witness, and a work tape. Only the work tape is limited to have space S. The input tape is read only, but can be read many times. The witness tape can have T symbols on it, but must be read sequentially and each symbol can only be read once. The witness can be thought of as being separated into d segments, each with a different quantifier. The change of quantifier is called an alternation. For example, nondeterministic algorithms have $d = 1$ since they only use existential quantifiers.

Let $\mathbf{ITIME}[T_V, T_P]$ be the set of languages with an interactive proof whose verifier runs in time T_V and

[1] By [Für+89], probability 1 can always be achieved, but that reduction has a significant cost to the prover's runtime.

[2] Throughout this work we mostly optimize for verification time and leave the proving time as a secondary consideration. This is in contrast to *doubly efficient interactive proofs* (see [Gol18]) in which we insist on a polynomial-time prover. In this "doubly-efficient" regime, interactive proofs with a polynomial-time prover and almost linear time verifier are known for linear depth, poly-size, uniform circuits [GKR15] and poly-time and bounded-poly space computation [RRR16]. See Section 2 for details.

whose prover runs in time T_P. If T_P is omitted, we assume it is the trivial bound of $T_P = O(2^{T_V})$. In this paper, all our protocols are public-coin and have perfect completeness.

Our most general result is an interactive protocol for alternating bounded-space algorithms.

Theorem 1.1 (Interactive Proof For Alternating Space)**.** *For any T, S, and d constructible in time $O(S\log(T))$ and space $O(S)$:*

$$\mathbf{ATISP}_d[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}\left(n + S\log(T) + Sd\right), 2^{O(S)}\right].$$

Further, the verifier runs in space $O(S\log(d+S))$, the protocol is public coin, has $O(S\log(S)(\log(T)+d))$ rounds, $O(S\log(S)(\log(T)+d)\log(d+S))$ bits of communication, and perfect completeness.

For $d = O(\log(T))$, up to small polylog(S) factors, our protocol has the same verifier time and prover time as the best known protocol for deterministic bounded space algorithms [Coo22b]: verifier time $\tilde{O}(n+S\log(T))$ and prover time $2^{O(S)}$. As a special case for nondeterministic algorithms, this gives an interactive protocol with verifier time $\tilde{O}(n + S\log(T))$, improving upon the nondeterministic algorithms in [Coo22b], whose verifiers required time $\tilde{O}(n + S\log(T)^2)$, by a factor of $\log(T)$. We note $\log(T)$ may be as large as S.

In a limited sense, these results could be seen as tight, as they match, up to polylog(S) factors, the best known results for simulating deterministic algorithms by alternating ones. Chandra, Kozen, and Stockmeyer [CKS81] show that any deterministic algorithm running in time T and space S has an alternating algorithm running in time $S\log(T)$. Specifically, $\mathbf{TISP}[T,S] \subseteq \mathbf{ATISP}_{\log(T)}[O(S\log(T)), O(S)]$. If we improved our verifier time dependence on $S\log(T)$ or Sd, this would improve the time of alternating algorithms simulating deterministic ones.

For $d = T$, Theorem 1.1 improves over the best known interactive proofs for alternating algorithms, with unbounded alternations, by Fortnow and Lund [FL93], which have verifier time $\tilde{O}(n + S^2T)$ and verifier space $O(S\log(T))$. Our protocol's verifier is at least a factor S faster (when $ST = \Omega(n)$).

See Table 1 for a comparison of how our protocol compares to prior protocols for nondeterministic algorithms, and Table 2 for a comparison of how our protocol compares to prior protocols for alternating algorithms. See Section 2 for a more detailed comparison to prior works.

The best verifiers [Coo22b; Tha20] for deterministic algorithms have verifier time $\tilde{O}(S\log(T)+n)$, verifier space $O(S\log(S))$, and provers with time $2^{O(S)}$. The best provers [RRR16] for deterministic algorithms have prover time $T^{1+o(1)}\text{poly}(S)$, but require verifier time $T^{o(1)}\text{poly}(S) + n\text{polylog}(T)$. These protocols are incomparable for T much larger than S, but much smaller than 2^S.

$\mathbf{NTISP}[T,S]$	Verifier Time	Verifier Space	Prover Time
[Sha92]	$(n+S)\log(T)^2$	$(n+S)\log(n+T)$	$2^{\text{poly}(S,n)}$
[FL93]	$n + S^3\log(T)$	$S\log(S)$	$2^{\text{poly}(S,n)}$
[GKR15]	$n + S^2\log(T)$	$S\log(S)$	$2^{O(S)}$
[Coo22b]	$n + S\log(T)^2$	$S\log(T)$	$2^{O(S)}$
This Work	$n + S\log(T)$	$S\log(S)$	$2^{O(S)}$

Table 1: Comparison of different protocols for $\mathbf{NTISP}[T,S]$ with polylog(S) factors omitted.

When $S = O(\log(n))$, our prover runs in polynomial-time. This gives us doubly efficient proofs for alternating algorithms with few alternations and logarithmic space. As a special case, we give doubly efficient interactive proofs for **NL** where the number of bits communicated is $\tilde{O}(\log(n)^2)$. This improves on the amount of communication achieved by GKR (specialized for **NL**), which uses $\tilde{\Omega}(\log(n)^3)$ bits of communication.

Corollary 1.2 (Doubly Efficient Interactive Proofs for **NL**)**.** **NL** *has interactive protocols whose provers run in polynomial time, verifiers run in quasilinear time, verifiers use $\tilde{O}(\log(n))$ space, the protocol uses $\tilde{O}(\log(n)^2)$ rounds, $\tilde{O}(\log(n)^2)$ bits of communication, is public coin and has perfect completeness.*

$\mathbf{ATISP}_d[T,S]$	Verifier Time	Verifier Space	Prover Time
[Sha92]	$(n+S(\log(T)+d))S(\log(T)+d)$	$n+S\log(T)+Sd$	$2^{\text{poly}(S,n)}$
[FL93]	$n+S^2T$	$S\log(T)$	$2^{\text{poly}(S,n)}$
[GKR15]	$n+S^2\log(T)+S^2d$	$S\log(S+d)$	$2^{O(S)}$
[Coo22b]	$n+(S\log(T)+Sd)^2$	$S\log(T)+Sd$	$2^{O(S\log(T)+Sd)}$
This Work	$n+S\log(T)+Sd$	$S\log(S+d)$	$2^{O(S)}$

Table 2: Comparison of different protocols for $\mathbf{ATISP}_d[T,S]$ with polylog(S) factors omitted.

More generally, our protocols for nondeterministic algorithms use a factor $\log(T)$ less communication then the previous best protocols by Cook, and match the best prior protocols for deterministic algorithms, up to polylogarithmic factors.

Unbounded Fan in Circuit Results. Let $\mathbf{SIZE}-\mathbf{DEPTH}[2^S,d]$ be the set of space $O(S)$ uniform circuits of size 2^S and depth d with *unbounded* fan in AND and OR gates. Let T-uniform $\mathbf{SIZE}-\mathbf{DEPTH}[2^S,d]$ be the set of time T uniform, space S circuits of size 2^S and depth d with *unbounded* fan in AND and OR gates. Then due to a close relationship between alternating circuits and low depth circuits by Ruzzo and Tompa [SV84] (see Section 3.3), we have

Theorem 1.3 (Uniform Shallow Circuits Have Fast Interactive Proofs)**.** *For any d,T,S constructible in time $O(S\log(T))$ and space $O(S)$, we have*

$$T\text{-uniform } \mathbf{SIZE}-\mathbf{DEPTH}\left[2^S,d\right] \subseteq \mathbf{ITIME}\left[\tilde{O}(n+S\log(T)+Sd),2^{O(S)}\right]$$
$$\mathbf{SIZE}-\mathbf{DEPTH}\left[2^S,d\right] \subseteq \mathbf{ITIME}\left[\tilde{O}(n+S^2+Sd),2^{O(S)}\right].$$

Further, the verifier runs in space $O(S\log(d+S))$ and the protocol is public coin and has perfect completeness.

For fan in 2 circuits, this matches the verifier time of GKR, while the prover time remains polynomial in the circuit size[3]. For unbounded fan in circuits, or for alternating algorithms, our verifier is a factor of S faster than GKR.

1.2 Proof Overview

We start by reviewing our efficient interactive proofs for deterministic algorithms. Then we explain the difficulty of extending this to nondeterministic algorithms, and how to overcome these problems. Finally we show how to extend this technique to alternating algorithms. We assume familiarity with the sumcheck protocol [Lun+90]. For a more detailed explanation of our interactive proofs for deterministic algorithms, see [Coo22b], the nearly identical protocol by Thaler [Tha22, Section 4.5.5] (see also [Tha20]), the similar protocol by Rudich [BM00], or Hirsch, Melkebeek and Smal [HMS13].

1.2.1 Deterministic Algorithms

For a deterministic algorithm A, we first reduce the problem to repeated matrix squaring, then give an interactive protocol for that. Suppose A runs in time T on some input x and has unique start state a and accept state b. Let M be the adjacency matrix of A's computation graph on input x. Then A accepts x if and only if $(M^T)_{a,b}=1$ (where M^T is M raised to the Tth power, *not* M transposed). For notation, we write $M_{a,b}$ as $M(a,b)$. At a high level, the idea is that if we have an interactive protocol that can reduce a

[3]Note however that recent improvements [CMT12; Tha13; Xie+19; Zha+21] of GKR have a (close to) linear prover, whereas our prover is only polynomial in the circuit size.

claim that $M^{2i}(u,v) = \alpha$ to the claim that $M^i(u',v') = \alpha'$, then by applying this protocol $\log(T)$ times, we can verify the value of $M^T(a,b)$. We give such a reduction, but on the multilinear extensions of M^{2i} and M^i.

Like [Coo22b; Tha20], we reduce to matrix exponentiation and give an interactive protocol for that, instead of reducing to a quantified Boolean formula [Sha92], or to a uniform circuit [GKR15]. This both simplifies the protocol somewhat and makes it more efficient to compute the prover. The idea is to arithmetize these adjacency matrices, and then use a sum check [Lun+90] to reduce the statement about M^{2i} to the statement about M^i. In particular, we use the sum check for matrix exponentiation given by Thaler [Tha13], details follow.

For a finite field $\mathbb{F}$, for any i define $\widehat{M^i} : \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$ as the multilinear extension of M^i. That is, $\widehat{M^i}$ is multilinear and for each $u, v \in \{0,1\}^S$ we have $\widehat{M^i}(u,v) = M^i(u,v)$. Then observe that for any i, and $u, v \in \mathbb{F}^S$ we have

$$\widehat{M^{2i}}(u,v) = \sum_{w \in \{0,1\}^S} \widehat{M^i}(u,w)\widehat{M^i}(w,v).$$

To see that this formula is correct, first observe that it is correct for *Boolean* values as it precisely corresponds to the definition of matrix multiplication. So the formula is correct on Boolean values. Since both sides of the equation are multi-linear[4], it follows that the formula holds for all values.

Then, we can use the sum check of [Lun+90] to reduce this to a claim that for some $w' \in \mathbb{F}^S$ and some $\beta \in \mathbb{F}$ we have $\beta = \widehat{M^i}(u,w')\widehat{M^i}(w',v)$. Then using a multi-point reduction, as was done in GKR [GKR15], we reduce this to a claim that for some $u', v' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ we have that $\alpha' = \widehat{M^i}(u',v')$.

Finally running this $\log(T)$ times gives the interactive protocol for deterministic algorithms, since the verifier can efficiently calculate $\widehat{M}$ itself.

We remark that, using linearization type ideas (as in [She92]), the above can be extended from the task of deciding whether a deterministic algorithm accepts, to verifying the multilinear extension of a function that the algorithm computes. This will be important for us later on when we use the above interactive proof as a subroutine in the protocol for nondeterministic algorithms.

1.2.2 Nondeterministic Algorithms and Changing Arithmetization

To try to apply this technique to a nondeterministic algorithm, A, we immediately encounter an issue with how to formulate the problem. Namely, if we are doing arithmetic over $\mathbb{Z}$, if the underlying matrix M corresponds to a non-deterministic computation, then the matrix $M^T_{a,b}$ is no longer 1 if and only if A accepts x. Rather, $M^T_{a,b}$ specifies the number of length T paths from a to b. This might be as large as $2^{\Omega(T)}$. If we do arithmetic over a field of characteristic q, then $M^T_{a,b}$ is the number of paths mod q. If the number of paths is some adversarial product of many small primes, we may need $q = \Omega(T)$ for the number of accepting paths to be non zero, mod q. This gave the less efficient verifier time for nondeterministic algorithms in [Coo22b].

We will still solve this problem by arithmetization, but we need to change our matrix multiplication from a field matrix multiplication to a binary multiplication, then arithmetize that. We define the matrix multiplication with binary operations where multiplication is AND and addition is OR. So let $M^{(2)} : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ denote this binary matrix multiplication, squaring, so that for any $u, v \in \{0,1\}^S$ we have

$$M^{(2i)}(u,v) = \bigvee_{w \in \{0,1\}^S} M^{(i)}(u,w)M^{(i)}(w,v).$$

With this form of matrix exponentiation, it suffices to check if $M^{(T)}_{a,b} = 1$. To do so, we convert binary matrix multiplication into an algebraic circuit. The obvious approach is to use a formula like

$$\tilde{M^{(2i)}}(u,v) = 1 - \prod_{w \in \{0,1\}^S} \left(1 - \widehat{M^{(i)}}(u,w) \cdot \widehat{M^{(i)}}(w,v)\right).$$

[4] To show it is multilinear, we take any variable, say u_i, and show the formula is linear in u_i. For any w see that $\widehat{M^i}(u,w)$ is linear in u_i since $\widehat{M^i}$ is multilinear, and $\widehat{M^i}(w,v)$ is constant in u_i. Thus $\sum_{w \in \{0,1\}^S} \widehat{M^i}(u,w)\widehat{M^i}(w,v)$ is linear in u_i.

Unfortunately, this has too high of individual degree: 2^S. One can insert some linearization operations between the multiplications to reduce the degree, like those used by Shen [She92]. But then for each of the S variables in w, one would need to add $O(S)$ linearization operations, giving a size $O(S^2)$ algebraic circuit, which we cannot afford.

Instead, we use an idea of Goldreich and Rothblum [GR20] to probabilistically reduce the degree of these large conjunctions by leveraging the Razborov-Smolensky [Raz87; Smo87] approximation of large disjunctions as low degree polynomials. Razborov-Smolensky give a reduction from a large disjunction to a random parity check that succeeds with high probability:

$$\forall g \in \{0,1\}^n : \Pr_{r \in \{0,1\}^n}\left[\bigvee_{i\in[n]} g_i = \sum_{i\in[n]} g_i r_i \pmod 2\right] \geq \frac{1}{2}.$$

We note that if $g = 0^n$, then for any r, we have $\sum_{i\in[n]} g_i r_i \pmod 2 = 0$. That is, the error is one sided. The formula $\sum_{i\in[n]} g_i r_i \pmod 2$ is a linear polynomial in a field of characteristic 2. As this is useful for us, we shall only work with fields of characteristic 2 in this paper.

Then, taking an OR of k independent choices of randomness, we get an individual degree k polynomial that succeeds with probability $1 - \frac{1}{2^k}$. If $n = 2^S$ and $k = S$, this gives us a degree $\log(n)$ polynomial for the disjunction that is only wrong with probability $\frac{1}{n}$.

The idea is to replace our boolean formula with a low degree polynomial through Razborov-Smolensky. So let $D_r : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ be a function outputting our random bits. Here $2^\ell = k = O(S)$ is the number of choices of random bits. Then our new approximation for $M^{(2i)}$ is

$$\widetilde{M^{(2i)}}(u,v) = 1 - \prod_{j\in\{0,1\}^\ell}\left(1 - \sum_{w\in\{0,1\}^S} D_r(j,w)\widehat{M^{(i)}}(u,w)\widehat{M^{(i)}}(w,v)\right). \tag{1}$$

Now we only need to insert $\ell = \log(S)$ levels of linearizations. In the technical details of the paper, we will not actually use algebraic circuits with linearization operations, but will work with these polynomials directly with an "unlinearization" procedure, to avoid discussing circuit uniformity.

1.2.3 Efficient Randomness

At this point we encounter a problem - Eq. (1) calls for sampling $2^{\ell+S}$ random bits for D_r, which we cannot afford (since we want our verifier to run in time $\tilde{O}(S)$). So as in GR, we need to sample these using an ϵ biased set. For our ϵ biased set, we use the same one as GR, described in [Alo+90] (which is based on a Reed-Solomon code concatenated with a Hadamard code).

Thus, for every value of $j \in \{0,1\}^\ell$, we would like to set $D_r(j,\cdot)$ to be an ϵ-biased set. As $\ell = \log(S)$, if we were to sample these independently, as in GR (i.e., the protocol given in [GR20]), our verifier would require $O(S^2)$ bits of randomness. Instead, we sample these small bias sets in a correlated manner - via a random walk on an expander (each node in the expander specifies a seed for a small bias set). We use the Margulis [Mar73] expander since it is a constant degree, constant spectral expander with extremely simple edge descriptions: simple additions and subtractions. This makes it very easy to take a start vertex and a (specification of a) random walk and compute any given step on that walk in both small space and small time.

Thus, we only require $R = O(S)$ truly random bits to describe a length $O(S)$ random walk on the ϵ biased sets described by a Reed-Solomon code concatenated with a Hadamard code. So let $D : \{0,1\}^R \times \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ be a function that generates our pseudorandomness, given R bits of true randomness. The verifier first chooses that randomness r, and then $D_r(j,w) = D(r,j,w)$.

Since D is both space and time efficient, we can have the prover compute its value for the verifier, and then have the verifier run the *deterministic* interactive protocol to confirm its value. In contrast, the GR verifier must calculate some low degree extension of D_r directly to use a constant number of rounds. This saves us time over GR.

Finally, as in GR, there is a chance that our pseudorandom bits give a polynomial that fails to compute the disjunctions correctly. In this case, to get perfect completeness we need to prove that the pseudorandom bits are incorrect. To do this, the prover just finds a disjunction closest to the input where the low degree approximation fails and tells the verifier where it fails. This would be a gate where its value in the low degree polynomial is one thing, but one of its input gates should force it to be something else. For instance, an OR gate with a value of 0, and an input to it with a value of 1. Then the verifier can run the interactive protocol to confirm that the low degree polynomial indeed says the gate's value conflicts with its input gate value, showing the pseudorandom bits were bad.

1.2.4 Alternating Algorithms In Terms Of Nondeterministic Algorithms

To use our protocol with alternating algorithms, we want to reduce the alternating algorithm to one with a few large disjunctions or conjunctions over a nondeterministic algorithm. This is similar to what is done when converting alternating algorithms to alternating circuits. Once we have few conjunctions and disjunctions over a nondeterministic algorithm, we can do the same low degree approximations again.

The idea is to, instead of quantifying over the symbols in the read once proof, quantify over the potential states the algorithm could be in when the quantifier changes. Then a nondeterministic algorithm describes if a proof could cause the state to change from one intermediate state to the next when the quantifier changes.

For example, suppose A is an algorithm with $d = 2$ alternations and running in space S recognizing language L. Think of A as a deterministic algorithm taking a proof and outputting true or false. Then since A is an alternating algorithm, $x \in L$ if and only if

$$\forall \text{BigProof1} : \exists \text{BigProof2} : A(x, (\text{BigProof1}, \text{BigProof2})).$$

We can instead be more fine grain with A and talk about its states. Let a be the start state of A and b be its unique accept state. Let B be the algorithm which takes an initial state u a final state v and a proof p, then checks if A starting at u is at state v when given the proof p after time $|p|$. Then our algorithm accepts x if and only if

$$\forall w \in \{0,1\}^S : ((\exists \text{Proof1} : B(x, a, w, \text{Proof1})) \implies (\exists \text{Proof2} : B(x, w, b, \text{Proof2}))).$$

If we know how long Proof1 is supposed to be, we can replace

$$\exists \text{Proof1} : B(x, a, w, \text{Proof1})$$

with a nondeterministic algorithm C. Then our alternating algorithm becomes

$$\forall w \in \{0,1\}^S : C(x, a, w) \implies C(x, w, b).$$

Now, beside our nondeterministic algorithm, we are only quantifying over a variable of size $O(S)$, whereas Proof1 has size $O(T)$.

For a more general example, we can replace

$$\forall \pi_1 : \exists \pi_2 : \forall \pi_3 : \exists \pi_4 : A(x, (\pi_1, \pi_2, \pi_3, \pi_4))$$

with

$$\forall w_1 : C(x, a, w_1) \implies (\exists w_2 : C(x, w_1, w_2) \wedge (\forall w_3 : C(x, w_2, w_3) \implies C(x, w_3, b))).$$

1.2.5 Protocols for Alternating Algorithms

At this point, each quantification is now only over S variables, so we can use the same trick as before to replace these quantifications with low degree polynomials. Each of the universal quantifiers gets replaced with a large conjunction, and each of the existential quantifiers gets replaced with a large disjunction. Then we use Razborov-Smolensky to replace these conjunctions and disjunctions with low degree polynomials and use an interactive proof to remove the quantifiers one by one.

A few subtleties show up when doing this. One subtlety of this process is that in a straightforward reduction, a d alternation algorithm would give our verifier a claim about C at d different places. Running an interactive protocol d times to confirm each of these d claims independently would require time $dS\log(T)$, which is too much for us. Instead, we need to use a multi point reduction again to reduce this to a claim about C at one location before running an interactive protocol to confirm that value.

Another subtlety is that it is not convenient to represent C as a nondeterministic algorithm taking two states as an input and checking if there is a computation path from one to the other. It is more convenient to describe C directly with the computation graph of A (now viewing A as a nondeterministic algorithm). For this to work, we need to make sure each alternation takes the same amount of time, say T. Then we write $C(x,u,v) = \widehat{M_x^{(T)}}(u,v)$.

So for example, consider the simple case of a 2 alternation algorithm. That is, suppose we want to verify that

$$\forall w \in \{0,1\}^S : C(x,a,w) \implies C(x,w,b).$$

As described before, we replace C with $\widehat{M^{(T)}}$. So we want to verify

$$\forall w \in \{0,1\}^S : \widehat{M_x^{(T)}}(a,w) \implies \widehat{M_x^{(T)}}(w,b).$$

Now we need to arithmetize the formula being quantified. So let

$$\widetilde{E}(w) = 1 - \widehat{M_x^{(T)}}(a,w)(1 - \widehat{M_x^{(T)}}(w,b)).$$

See that E is low degree and agrees with the predicate $\widehat{M_x^{(T)}}(a,w) \implies \widehat{M_x^{(T)}}(w,b)$ on binary inputs. Of course, $\tilde{E}$ is not multilinear, it has individual degree 2. Luckily, if we let $\widehat{E}$ be the multilinear function consistent with D on binary inputs, then one can use an unlinearization operation (similar to those used by Shen [She92]) to reduce from a statement about $\widehat{E}$ to a statement about $\widetilde{E}$. So we need to verify that

$$\forall w \in \{0,1\}^S : \widehat{E}(w).$$

Using our low degree approximation, our verifier first chooses D_r, then wants to check if

$$1 = \prod_{j \in \{0,1\}^\ell} \left(1 - \sum_{w \in \{0,1\}^S} D_r(j,w)(1 - \widehat{E}(w))\right). \tag{2}$$

Then we can reduce this to a statement about $\widehat{D_r}$ at a random location, and $\widehat{E}$ at a random location by using $\ell = O(\log(S))$ product reductions. We can unlinearize the statement about $\widehat{E}$ to get a claim about $\widetilde{E}$, or equivalently, about $\widehat{M_x^{(T)}}$ at two locations. Now we can verify the value of $\widehat{M_x^{(T)}}$ by using our protocol for nondeterministic algorithms. But to avoid doing this twice, we first run a multi-point reduction to reduce this to a statement about $\widehat{M_x^{(T)}}$ at one location first.

We can do a similar thing d times for an alternating algorithm. One more subtlety is that for $d > 2$, we need to make $\widehat{E}$ a function of a and b. This is so that we can view the formula in Eq. (2) as a function of a and b so we can properly linearize and unlinearize it with respect to a and b. See the full proof for details.

Remark (Proof For Unbounded Fan in Depth Circuits Directly)**.** *We could have made an interactive protocol for unbounded fan-in circuits directly. After all, we start with a formula that is essentially the low depth, unbounded fan in circuit for an alternating algorithm, if we view C as a low depth circuit. We can think of our alternating algorithms as a particular kind of very uniform circuit. We don't give an interactive proof for circuits directly to avoid handling uniformity.*

One reason we chose not to just provide an interactive protocol for circuits directly is that we need a faster interactive protocol for deterministic algorithms as a subroutine to verify our pseudorandom bits. Since we view this interactive protocol as a problem for bounded space, we find it natural to present the rest of the results in this framework.

1.2.6 Extensions

We note that our paper focuses on verifier time, so we have not optimized other parameters, like verifier space. There are also some other straightforward extensions to further generalizations of alternations we don't prove here.

Remark (Parity Gates and Parity Quantifiers)**.** *Like GR, our techniques can also be used on alternating circuits with parity gates, or bounded space algorithms with parity quantifiers. This is clear since a parity gate is an addition gate over fields of characteristic 2, so is of low degree already.*

We emphasize that our protocol is different from GR since we need more randomness efficient pseudorandom bits, efficient computation of those pseudorandom bits, more rounds of interaction to keep our degree (and thus verifier time) lower, use of an interactive protocol for deterministic algorithms as a subroutine, and by using connections between low space algorithms, low alternation algorithms, and uniform, low depth circuits.

Remark (Space Efficiency of Our Verifier)**.** *While we only achieve a verifier running in space $O(S\log(S))$, for any $\epsilon > 0$ we should be able to get verifier space $O(S/\epsilon)$ using standard techniques, at the cost of increasing verifier time by a factor of S^{ϵ}. Specifically, instead of using multilinear polynomials, we would use individual degree S^{ϵ} polynomials. Since we are focused on improving verifier time, we do not prove this result.*

This technique was used by Shamir, Fortnow and Lund, and GKR [Sha92; FL93; GKR15] to give the space efficiency claimed in those papers. We state the special case where $\epsilon = \frac{1}{\log(S)}$ in our results since we want to compare verifier time.

2 Previous Results

In this section we give a detailed summary of prior interactive proofs in the literature. Most of these results are summarized in Tables 1 and 2 above, and so this section can be safely skipped by an impatient reader.

The first interactive proofs for **PSPACE** were originally given by Shamir [Sha92] using techniques by Lund, Fortnow, Karloff, and Nisan [Lun+90]. Shamir did not analyze the verifier time and prover time beyond showing that the verifier runs in time $\text{poly}(S)$ for a space S algorithm. However, analyzing the space efficient interactive protocol included in Shamir's result, we see that the protocol runs in time $\tilde{O}((n+S)\log(T))$, before the state transition function of the algorithm is arithmetized. If one assumes we know a linear sized Boolean formula for the state transition function, then we get:

Theorem 2.1 (Shamir's Protocol (Assuming Good Arithmetization))**.** *Let S and T be time $O(S\log(T))$ and space $O(S)$ computable with $S = \Omega(\log(n))$. Then*

$$\begin{aligned}\mathbf{TISP}[T,S] \subseteq &\mathbf{ITIME}\left[\tilde{O}((n+S)\log(T))\right]\\ \mathbf{NTISP}[T,S] \subseteq &\mathbf{ITIME}\left[\tilde{O}((n+S)\log(T)^2)\right].\end{aligned}$$

Further the verifier space for deterministic algorithms is $O((n+S)\log(n+S))$, and for nondeterministic algorithms is $O((n+S)\log(n+T))$.

And as a corollary, using the relationship between alternations and space, Theorem 3.8, with Shamir's IP gives

Corollary 2.2 (Shamir's Protocol For **ATISP**)**.** *Let S and T be time $O(S\log(T))$ and space $O(S)$ computable with $S = \Omega(\log(n))$. Then*

$$\mathbf{ATISP}_d[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}((n+S\log(T)+Sd)(S\log(T)+Sd))\right].$$

Further the verifier space is $O((n+S\log(T)+Sd)\log(n+Sd\log(T)))$.

In a followup work, Fortnow and Lund [FL93] gave an interactive proof for bounded time and space alternating algorithms. Interactive protocols can also be efficiently converted to alternating algorithms. So this showed a strong relationship between time and space bounded alternating algorithms and verifier time and space bounded interactive protocols. They proved[5]:

Theorem 2.3 (Fortnow and Lunds Protocol)**.** *Let S and T be time $O(S^2T + n)$ and space $O(S\log(T))$ computable with $S = \Omega(\log(n))$. Then*

$$\mathbf{ATISP}[T, S] \subseteq \mathbf{ITIME}\left[\tilde{O}(S^2T + n)\right].$$

Further the verifier runs in space $O(S\log(T))$.

Although Fortnow and Lund's protocol only handles the case where $d = T$, when $d \ll T$, we can apply Theorem 3.9 to get

Theorem 2.4 (Fortnow And Lunds Protocol When $d \ll T$)**.** *Let S and T be time $O(S^2T + n)$ and space $O(S\log(T))$ computable with $S = \Omega(\log(n))$. Then*

$$\mathbf{ATISP}_d[T, S] \subseteq \mathbf{ITIME}\left[\tilde{O}(S^3(d + \log(T)) + n)\right].$$

Further the verifier runs in space $O(S\log(Sd))$.

In a very influential paper, Goldwasser, Kalai and Rothblum gave doubly efficient interactive proofs for depth bounded circuits with *fan in 2* [GKR15] (GKR). To distinguish fan in 2 circuits from the unbounded fan in circuits discussed earlier, let $\mathbf{SIZE} - \mathbf{DEPTH}_2[2^S, d]$ denote space S uniform *fan in 2* circuits of size 2^S and depth d. Doubly efficient proofs are proofs where the prover runs in polynomial time and the verifier runs in almost linear time. With improvements by Cormode, Mitzenmacher, and Thaler [CMT12], GKR gives:

Theorem 2.5 (GKR For Depth)**.** *Given S and d are computable in time $O(n + Sd)$ and space $O(S)$. Then*

$$O(S) - \text{uniform } \mathbf{SIZE} - \mathbf{DEPTH}_2[2^S, d] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + dS), \tilde{O}(2^S)\right].$$

Further the verifier uses space $O(S\log(S + d))$, the protocol has $O(dS)$ rounds and uses $O(dS\log(d + S))$ bits of communication.

We emphasize that GKR is for *fan in 2* circuits, but our protocols are for *unbounded* fan in circuits. One can use Theorem 3.14 with GKR to get protocols for log space uniform computation. Much work has been done to further improve the low order terms in GKR style proofs [Xie+19; Zha+21].

Using the relationship between alternating algorithms and circuits Lemma 3.17 gives:

Corollary 2.6 (GKR For Alternating Time Space)**.** *Given T, S and d are computable in time $O(n + S^2d)$ and space $O(S)$. Then*

$$\mathbf{ATISP}_d[T, S] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + S^2\log(T) + S^2d), 2^{O(S)}\right].$$

Further the verifier uses space $O(S\log(S + d))$, the protocol has $O(S^2(\log(T) + d))$ rounds and uses $O(S^2(\log(T) + d)\log(d + S))$ bits of communication.

A work by Cook [Coo22b] gave an interactive protocol for space bounded computation which improved the verifier time over Shamir's protocol when $S = o(n)$ while giving a smaller prover time. For deterministic algorithms, a nearly identical protocol was given by Thaler [Tha20], but with a worse dependence on n due to a worse arithmetization. A protocol by Rudich [BM00], with subsequent improvements by Hirsch, Melkebeek and Alexander [HMS13] is also nearly identical, but also with a worse arithmetization than Cook's.

[5]Fortnow and Lund state the verifier time as $\tilde{O}((S^2T + n)S^\epsilon)$ and space as $O(S\log(T)/\log(S))$ for any $\epsilon > 0$. But we state the result when ϵ is very small for a more direct comparison.

Theorem 2.7 (Cook's Protocol)**.** *Let S and T be time $O(S\log(T))$ and space $O(S)$ computable with $S = \Omega(\log(n))$. Then*

$$\mathbf{TISP}[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + S\log(T)), 2^{O(S)}\right]$$
$$\mathbf{NTISP}[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + S\log(T)^2), 2^{O(S)}\right].$$

For deterministic algorithms, the verifier space is $O(S\log(S))$, the number of rounds is $O(S\log(T))$, the number of bits communicated is $O(S\log(T)\log(S))$, and the protocol is public coin and has perfect completeness.

For nondeterministic algorithms, the verifier space is $O(S\log(T))$, the number of rounds is $O(S\log(T))$, the number of bits communicated is $O(S\log(T)^2)$, and the protocol is public coin and has perfect completeness.

Again, we can use Theorem 3.8 to get

Corollary 2.8 (Cook's Protocol For **ATISP**)**.** *Let S and T be time $O(S\log(T))$ and space $O(S)$ computable with $S = \Omega(\log(n))$. Then*

$$\mathbf{ATISP}_d[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + (S\log(T) + Sd)^2), 2^{O(S\log(T)+Sd)}\right].$$

Further the verifier space is $O((S\log(T) + Sd)\log(Sd\log(T)))$.

The GKR protocol, as well as ours, only have polynomial time provers when $S = O(\log(S))$. Reingold, Rothblum and Rothblum [RRR16] (RRR) gave a doubly efficient protocol whose prover is polynomial as long as T is polynomial, but the verifier is only linear if T is polynomial.

Theorem 2.9 (RRR Protocol for Deterministic)**.** *For any constant $\delta > 0$, and integers S and T computable in time $T^{O(\delta)}S^2$, and $T = \Omega(n)$ we have*

$$\mathbf{TISP}[T,S] \subseteq \mathbf{ITIME}\left[O(n\text{polylog}(T) + T^{O(\delta)}\text{poly}(S)), T^{1+O(\delta)}\text{poly}(S)\right].$$

Further the protocol only has $\left(\frac{1}{\delta}\right)^{O(1/\delta)}$ rounds.

We note the result in the [RRR16] paper allows sub constant δ, but is complex and can not give a verifier with time $\text{poly}(\log(T)S)$. Unfortunately, the algorithm time blow up from something like Theorem 3.8 is too much for RRR to handle and gives exponential time verifiers.

A major limitation of the RRR protocol is that the number of rounds are $\left(\frac{1}{\delta}\right)^{O(1/\delta)}$, but one might expect that it would be possible to achieve $O\left(\frac{1}{\delta}\right)$ rounds. Goldreich and Rothblum (GR) [GR20] gave an interactive protocol that achieves this, but only for very shallow circuits: $\mathbf{AC}^0[\oplus]$.

Theorem 2.10 (GR Protocol For $\mathbf{AC}^0$ with Parity)**.** *For constants c, d and suppose that L is computable by time $O(\log(n))$ uniform circuits with unbounded fan in, AND, OR, parity and negation gates, depth d and size n^c. Then for every $\delta \in (0,1]$, then there is an interactive protocol for L with $O(cd/\delta)$ rounds, a time $O(n^{1+o(1)})$ verifier and a time $O(n^{c+o(1)})$ prover with only n^δ bits of communication.*

GR also gave an interactive protocol with fewer rounds for $\mathbf{NC}^1$.

Theorem 2.11 (GR Protocol For $\mathbf{NC}^1$)**.** *Suppose that L is computable by time $O(\log(n))$ uniform circuits with fan-in two and depth $O(\log(n))$. Then for every $\delta \in (0,1]$, then there is an interactive protocol for L with $O(1/\delta^2)$ rounds, a time $O(n^{1+o(1)})$ verifier and a time $\text{poly}(n)$ prover with only $n^{\delta+o(1)}$ bits of communication.*

While GR's protocols are highly optimized to the special case of very low depth circuits to get an interactive protocol with very few rounds, we use some of its techniques in this paper. Unfortunately, we can't reduce general space bounded computation to an $\mathbf{NC}^1$ circuit, so its unclear how to compare it to our results.

Other proofs that $\mathbf{IP} = \mathbf{PSPACE}$ worth mentioning are Shen 's [She92] variation of Shamir's proof and Meir's combinatorial proof that $\mathbf{IP} = \mathbf{PSPACE}$ [Mei13]. Neither of these improve verifier time.

3 Preliminaries

We assume the reader is familiar with basic complexity concepts like circuits, Turing machines, and big O notation. See [AB09] for a reference. For notation, we define $\tilde{O}$ to hide polylogarithmic factors in whatever is inside it in general, not specifically polylog(T) or polylog(n). That is:

Definition 3.1 (Big Tilde O)**.** *For functions $f, g : \mathbb{N} \to \mathbb{N}$, we define $f(n) = \tilde{O}(g(n))$ if and only if there exists some constant c such that $f(n) = O(g(n)\log(g(n))^c)$.*

3.1 Bounded Space and Alternating Algorithms

We denote by $\mathbf{TISP}[T, S]$ languages that are computable by a Turing Machine running in time T and space S.

Definition 3.2 (**TISP**)**.** *For functions $T, S : \mathbb{N} \to \mathbb{N}$, we say language L is in $\mathbf{TISP}[T, S]$ if there is an Turing Machine, A, running in time T and space S that decides L.*

We want interactive proofs for a generalization of nondeterministic algorithms called alternating algorithms, as was formally defined in [CKS81]. In a nondeterministic Turing Machine, from some Turing Machine states, there are multiple state transitions, and the algorithm accepts if there are any choice of state transitions leading to an accept state. That is, the Turing Machine accepts a state if there exists a transition out of that state to a state the Turing Machine accepts. We call these existential states with existential quantifiers, or exists quantifiers. Similarly, a co-nondeterministic algorithm is one with universal states such that the algorithm would accept on every state transition. We call these universal states with universal quantifiers, or for all quantifiers.

An alternating Turing Machine is like a nondeterministic one, except that it has both existential states and can use exists quantifiers, and also has universal states and can use for all quantifiers. These are called alternating machines because they can alternate between quantifiers, unlike nondeterministic algorithms or co-nondeterministic algorithms that must use only one. So one can define alternating **P**, **AP**, as polynomial time on an alternating Turing Machine. The natural complete language for alternating languages are quantified Boolean formulas. This is why $\mathbf{AP} = \mathbf{PSPACE}$. We naturally define $\mathbf{ATISP}[T, S]$ as the languages that are computable by a time T, space S alternating Turing Machine.

We further parameterize our alternating algorithms by the number of alternations. The number of alternations is the number of quantifiers it uses, or equivlanetly, the number of times it switches between existential and universal states, plus one. For instance, nondeterministic algorithms can be viewed as having one alternation, the second level of the polynomial hierarchy has two, and so on.

Definition 3.3 (**ATISP**)**.** *For functions $T, S, d : \mathbb{N} \to \mathbb{N}$, we say a language L is in $\mathbf{ATISP}_d[T, S]$ if there is an alternating Turing Machine, A, running in time T and space S that recognizes L such that on any input x, our algorithm A only changes from no quantification to one, from existential to universal quantifiers, or from universal to existential quantifiers d times.*

So for instance, nondeterministic time T and space S algorithms would be contained in $\mathbf{ATISP}_1[T, S]$.

3.2 Interactive Proofs

An interactive proof informally is a proof system where a verifier with access to unpredictable randomness can verify a result such that if the statement is true, an honest prover can convince the verifier with high probability. In this paper, honest provers succeed with probability 1. And if the statement is false, no prover, no matter how powerful, can convince the verifier the statement is true above some constant probability, as long as the prover can't predict the random bits of the verifier. In this paper, we focus on the perfect completeness case, where the prover can always convince the verifier of a true statement.

While we will mostly discuss the concept of an interactive protocol intuitively, we will briefly formally define an interaction between a prover and verifier for concreteness.

Definition 3.4 (Interaction Between Verifier and Prover (Int))**.** *Let V be an algorithm with access to randomness, that can make oracle queries, and outputs something in $\{0,1\}$. Let P' be any function, and x be an input.*

Now we define the interaction of V and P' on input x. For all i, define y_i to be V's ith oracle query given its first $i-1$ queries were answered with $z_1,\ldots,z_{i-1}$ and define $z_i = P'(x,y_1,\ldots,y_i)$. Then the y_i will be the verifiers messages (which will depend on some randomness used by the verifier), and the z_i will be the prover messages.

Define the output of V when interacting with P', $\mathrm{Int}(V,P',x)$, as the output of V on input x when its oracle queries are answered by $z_1,z_2,\ldots$. The verifier time and space will be the time and space of V. The number of rounds is the max number of messages the verifier might send to the prover.

Our protocols in this paper will all be public coin, so the messages $y_1,\ldots,y_i$ will just be the randomness used by the verifier.

Now we define interactive time. We note that in all our protocols, we achieve perfect completeness. That is, $c=1$.

Definition 3.5 (Interactive Time (**ITIME**))**.** *If for any language L, soundness $s \in [0,1]$, completeness $c \in [0,1]$, verifier V and prover P we have*

Completeness: *If $x \in L$, then $\Pr[\mathrm{Int}(V,P,x)=1] \geq c$, and*

Soundness: *if $x \notin L$, then for any function P' we have $\Pr[\mathrm{Int}(V,P',x)=1] \leq s$,*

then we say V and P are an interactive protocol for L with soundness s and completeness c.

If in addition verifier V runs in time T_V, soundness $s < \frac{1}{3}$, and completeness $c > \frac{2}{3}$, then

$$L \in \mathbf{ITIME}[T_V].$$

If P is also computable by an algorithm running in time T_P, we say

$$L \in \mathbf{ITIME}[T_V, T_P].$$

3.3 Relations Between Alternating and Other Complexities

We now outline the known relationships between alternating algorithms and other complexity classes.

The relationship between the polynomial hierarchy and **AC** circuits was used by [FSS81] to show oracle separations for the polynomial hierarchy. Chandra, Kozen, and Stockmeyer [CKS81] defined alternating algorithms and showed many of the foundational relationships between it and other complexity classes. They gave an equivalence between constant alternation alternating algorithms and the polynomial hierarchy, citing [Wra76].

Theorem 3.6 ([CKS81], Theorem 4.1)**.** *For any constant d,*

$$\mathbf{ATIME}_d[\mathrm{poly}(n)] = \Sigma_k P \cup \Pi_k P.$$

[CKS81] also showed many very strong relationships between alternating time and deterministic space, as well as alternating space and deterministic time.

Theorem 3.7 ([CKS81], Theorems 3.1-3.6)**.** *For $S = \Omega(n)$ and $T = \Omega(n)$:*

$$\begin{aligned}
\mathbf{NSPACE}[S] &\subseteq \mathbf{ATIME}[O(S^2)] \\
\mathbf{ATIME}[T] &\subseteq \mathbf{SPACE}[O(T)] \\
\mathbf{PSPACE} &= \mathbf{AP} \\
\mathbf{EXPSPACE} &= \mathbf{AEXP}.
\end{aligned}$$

For $S = \Omega(\log(n))$,

$$\begin{aligned}\mathbf{TIME}[T] \subseteq &\mathbf{ASPACE}[O(\log(T))]\\ \mathbf{ASPACE}[S] \subseteq &\mathbf{TIME}[2^{O(S)}]\\ \mathbf{P} = &\mathbf{AL}\\ \mathbf{EXP} = &\mathbf{APSPACE}.\end{aligned}$$

[CKS81] showed an extension of Savitch's theorem to alternating algorithms with few alternations.

Theorem 3.8 ([CKS81], Theorem 4.2 (credited to Borodin.))**.** *For* $S = \Omega(\log(n))$ *and* $T = \Omega(n)$*:*

$$\mathbf{ATISP}_d[T, S] \subseteq \mathbf{SPACE}[O(S\log(T) + Sd)].$$

This characterization was further generalized by Ruzzo [Ruz81] to give

Theorem 3.9 ([Ruz81], Proposition 1)**.** *For* $S = \Omega(\log(n))$ *and* $T = \Omega(n)$*:*

$$\mathbf{ATISP}_d[T, S] \subseteq \mathbf{ATISP}[O(S\log(T) + Sd), O(S)].$$

Remark. *[CKS81; Ruz81] actually states this theorem for* $\mathbf{ATISP}_d[2^S, S]$*, but it is clear from the proof that the* S^2 *term only needs the time to run Savitch's theorem, which is only space* $S\log(T)$*.*

Now we describe the relationship between alternating algorithms and bounded depth circuits. First, we give our notation for bounded depth circuits.

Definition 3.10 (Size, Depth, Uniformity of Circuits)**.** *For any* d*,* $S = \Omega(\log(n))$*,* $f \in \mathbb{N}$*, and uniformity measure* U^* *define*

$$U^* - \mathbf{SIZE} - \mathbf{DEPTH}_f[2^S, d]$$

to be the class of U^* *uniform circuits with fan in* f*, size* 2^S*, and depth* d *where the gate set is AND, OR, and NOT.*

We let $\mathbf{SIZE} - \mathbf{DEPTH}[2^S, d]$ *denote unbounded fan in, log space (space* $O(S)$*) uniform circuits of size* 2^S *and depth* d*. That is, the entire circuit description can be constructed in* $O(S)$ *space. This is* U_{BC} *in [Ruz81] and the uniformity used in [GKR15].*

For any circuit C *with* 2^S *gates, let* $G : \{0,1\}^S \to \{\wedge, \vee, \neq\}$ *be the function that takes a gate's index and outputs its type. Let* $D : \{0,1\}^S \times [f] \to \{0,1\}^S$ *be the function such that* $D(u, v)$ *is one if and only if* u *is an input to* v*. For* $T = \Omega(\log(n))$*, we let*

$$T\text{-uniform } \mathbf{SIZE} - \mathbf{DEPTH}_f[2^S, d]$$

be the circuits with fan in f*, size* 2^S*, and depth* d *such that* D *and* G *can be computed in time* T *and space* $O(S)$*. If* $T = O(S)$*, this is* U_D *in [Raz87] and similar to deterministic log time uniform [Bus87; MBIS90].*

Ruzzo [Ruz81] established a very strong relationship between very uniform, depth bounded circuits and time bounded alternating algorithms. As well as giving strong relationships between the different notions of uniformity, like those used by Borodin and Cook [Bor77; Coo79]. For the strongest relationships between bounded depth circuits and bounded alternating time, Ruzzo gave some very strong notions of uniformity.

Definition 3.11 ([Ruz81], U_E uniform.)**.** *We say a circuit family of size* 2^S *is* U_E *uniform if given any gate,* g*, and path* p *(with* $|p| \leq S$*) out of* g *(given as a sequence of left or right input gates), there is an algorithm running in time* $O(S)$ *who can verify if a given gate* y *is the gate reached by taking* p *out of* g*.*

Remark. *This allows us to traverse the circuit by only keeping track of our current path from a given gate, instead of updating our entire gates label at each step. Then whenever we want to actually get our current gate label, we can do so quickly given the path.*

This kind of uniformity is much more strict than the standard log space uniformity used by [GKR15], or even deterministic log time uniformity. However, standard Turing Machine to circuit reductions often have this kind of uniformity, and it allows Ruzzo to make a much stronger relationship between bounded depth circuits and alternating algorithms.

Theorem 3.12 ([Ruz81], Theorem 3). *For $S, T = \Omega(\log(n))$ both computable in time $O(S)$:*

$$\mathbf{ATISP}[T, S] \subseteq \mathrm{U}_E - \mathbf{SIZE} - \mathbf{DEPTH}_2[2^{O(S)}, T].$$

and

Theorem 3.13 ([Ruz81], Theorem 4). *For $S, T = \Omega(\log(n))$:*

$$\mathrm{U}_E - \mathbf{SIZE} - \mathbf{DEPTH}_2[2^S, T] \subseteq \mathbf{ATISP}[O(T), O(S)].$$

These show that for sufficiently uniform circuits, bounded depth is nearly exactly equivalent to bounded alternations. But GKR applies to log space uniform circuits, not just U_E uniform ones. Fortunately, Ruzzo gave a reduction from log space uniformity (called U_{BC} uniformity in [Ruz81]) to U_E uniformity.

Theorem 3.14 ([Ruz81], Theorem 2). *For $S, d = \Omega(\log(n))$ both computable in time $O(S)$:*

$$\mathbf{SIZE} - \mathbf{DEPTH}_2[2^S, d] \subseteq \mathrm{U}_E - \mathbf{SIZE} - \mathbf{DEPTH}_2[2^{O(S)}, O(d + S^2)].$$

This result can be further refined to give a relationship between depth d circuits with unbounded fan-in and alternating algorithms with d alternations. Such a result was credited to Ruzzo and Tompa and described by Stockmeyere and Vishkin [SV84].

Lemma 3.15 (Log Space Uniform Circuits Can Be Made Log Time). *For $S = \Omega(\log(n))$ computable in time $O(S)$ and any d:*

$$\mathbf{SIZE} - \mathbf{DEPTH}[2^S, d] \subseteq O(S) - \text{uniform } - \mathbf{SIZE} - \mathbf{DEPTH}[2^{O(S)}, O(d + S)].$$

A straightforward reduction reduces time T uniform circuit to an alternating algorithm. This uses similar ideas to those used by Ruzzo.

Lemma 3.16 (Uniform Shallow Circuits have Few Alternations and Low Space). *For $S, T = \Omega(\log(n))$ and $d \geq 1$ all computable in time $O(S)$:*

$$T\text{-uniform } \mathbf{SIZE} - \mathbf{DEPTH}[2^S, d] \subseteq \mathbf{ATISP}_d[O(dS + T), O(S)]$$

Proof Sketch. For a time T space $O(S)$ uniform circuit family of size 2^S and depth d, let A be the space S algorithm for C which takes as input a gate in the circuit, and recognizes its input gates.

Then the idea is to use $\exists$ on the OR gates, and to use $\forall$ on the AND gates. Our algorithm accepts a gate if it is an OR gate and there exists an input gate, or it is an AND gate and for all input gates it accepts. More specifically, for an OR gate, we ask if there exists a gate we transition to where both we accept that gate and we transition to it. For an AND gate, we ask if for all gates, we either don't transition to it, or we accept it.

This can be implemented in two ways. Either for a gate, we check if the gate is reachable, then quantify over the next level of the circuit, or use another quantifier to check if they are equal. We use two quantifiers on every layer, except the last layer before the input. On the last layer, we just check if the input bit is one, and if it is an input to this gate.

The quantifiers added between the layers of the circuit 'collapse' since they are next to quantifiers of the same kind. So we only need d quantifiers. The time is just the time to guess the sequence of gates plus the time to verify one of them (as each computation path verifies exactly one adjacency). Since we go through d gates of size S, the time for the sequence of gates is just $O(Sd)$. Thus the total time is just $O(Sd + T)$. For space, we only ever need to know the current gate, and a candidate input gate we need to check. This only requires space $O(S)$. □

Lemma 3.15, Lemma 3.16 and Theorem 1.1 together give Theorem 1.3.

Another proof credited to Ruzzo and Tompa [SV84] also show that:

Lemma 3.17 (Few Alternations and Low Space have Uniform Shallow Circuits)**.** *For $S, T = \Omega(\log(n))$ and $d \geq 1$ all computable in time $O(S)$:*

$$\mathbf{ATISP}_d[T, S] \subseteq O(S) - \text{uniform } - \mathbf{SIZE} - \mathbf{DEPTH}[2^{O(S)}, d + \log(T)].$$

This shows how one can use GKR to give a proof for alternating algorithms as well. But to use GKR, one must first make the fan in constant, which increases the depth by a factor of S. This is why GKR has a verifier time which is a factor S slower than ours.

3.4 Expander Graphs

We will use expander graphs to create hitting samplers through the hitting properties of expanders [Kah95]. See [Vad12] for a more detailed review of expander graphs. We will assume some basic familiarity with graphs here, and review the details of expanders we need.

Definition 3.18 (Expander Graph)**.** *We say a graph G is an expander graph if its normalized adjacency matrix has second largest eigenvalue some constant less than 1. We call the second largest eigenvalue its spectral expansion.*

By normalized adjacency matrix, we mean the transition matrix of a uniform random walk on the graph.

We use an expander graph given by Margulis [Mar73] proven by Gabber and Galil [GG79]. We use this expander because it's simple structure makes it very clear that we can compute random walks on it in very little time and linear space.

Lemma 3.19 (Efficient Expander Graphs)**.** *For any square $n = m^2$, there exists an expander graph G with constant degree d and constant spectral expansion $\lambda < 1$.*

Let V be the vertex set of G, and $E : V \times [d] \to V$ be the edge function taking in a vertex, v, and the index of an edge, e, out of v, and outputting the other vertex incident to e. Then E can be computed in space $O(\log(n))$ and time $O(\log(n))$.

Proof. The proof that the graph is an expander is given by [GG79]. The proof that the edge function can be computed efficiently is clear from the definition. We view $V = \mathbb{Z}_m \times \mathbb{Z}_m$ as the pairs of integers mode m. Then E is defined by

$$\begin{aligned}
E((a,b),1) &= (a+1, b)\\
E((a,b),2) &= (a-1, b)\\
E((a,b),3) &= (a, b+1)\\
E((a,b),4) &= (a, b-1)\\
E((a,b),5) &= (a+b, b)\\
E((a,b),6) &= (a-b, b)\\
E((a,b),7) &= (a, b+a)\\
E((a,b),8) &= (a, b-a).
\end{aligned}$$

Each of these is a simple addition or subtraction, which is easy to calculate in linear time. □

The main lemma about expanders we will use is that random walks on expanders graphs hit any subset with high probability. This was originally proved by Kahale [Kah95].

Lemma 3.20 (Random Walks on Expanders Hit With High Probability)**.** *Let G be an expander graph with spectral expansion λ and vertex set V. For any length t, let W_t be the set of random walks on G with length*

t. That is $w \in W_t$ if and only if $w = (v_1, v_2, \ldots, v_t) \subset V^t$ such that for each $i \in [t-1]$ we there is an edge from v_i to v_{i+1}.

Then for any set $B \subseteq V$ with $\frac{|B|}{|G|} < \mu$, and any walk length t we have

$$\Pr_{(v_1,\ldots,v_t)\in W_t}\left[\bigwedge_{i\in[t]} v_i \in B\right] \leq \left(\mu + \lambda(1-\mu)\right)^t.$$

3.5 Arithmetization

A core technique of standard interactive proofs is called "arithmetization". Arithmetization is the process of converting some Boolean function, f, to a low degree formula over a larger field, $\mathbb{F}$, which agrees with f on Boolean inputs. By using things like the Schwartz-Zippel lemma, this gives us an error correcting code encoding of a function, which help us design interactive proofs. The main function we will be arithmetizing is the state transition of Turing Machines. So let us formally define the state of a Turing machine.

First, we define our Turing machine's state. We use multi-tape Turing Machines that have a read only input tape, and a working tape. Then our Turing machine's state will be the program state of the Turing machine, the contents of the work tape, and the locations of the tape heads. We further encode the tape in a convenient, binary format.

Definition 3.21 (Turing Machine State)**.** *Let A be a two tape Turing Machine with a read only input tape, and a work tape. Let $D = \{-1, 0, 1\}$ be the set of symbols indicating how we should move a tape.*

Suppose A has alphabet Σ, states Q, start state $a \in Q$, accept state $b \in Q$, and state transitions $\Lambda : (Q \times \Sigma \times \Sigma) \times (Q \times D \times D \times \Sigma)$.

Notably, we do not restrict our state transitions to be functions, A may be a nondeterministic algorithm with many state transitions, or no state transitions. We identify Q with $\{0,1\}^{k_1}$ for some constant k_1, adding states with no transitions in Λ if necessary. We do the same for Σ.

Suppose on any length n input, A only ever reads from or writes the positions in $[S']$, and reads the input on positions in $[n']$ where $n' = n + 2$. The positions on the tape may be appropriately indexed so that the first bit of the input is position one, and the first bit of the working tape is at $S'/2$.

Then we define a state of A on input x with $|x| = n$ as any tuple $s = (q, h, w, m) \in Q \times [n'] \times [S'] \times \Sigma^{S'}$. We identify $[n']$ and $[S']$ with binary strings so that for some $S = O(S' + \log(n))$ we have $s \in \{0,1\}^S$. We also assume that A has a unique start state, and a unique accept state if it accepts.

Remark. *Note that if $S' = \Omega(\log(n))$, then $S = O(S')$. Thus, we generally just refer to S as the space of the algorithm, and s as the state. We will call q the instruction state. We assume that all algorithms are defined by such a Turing Machine.*

Remark. *We note that without loss of generality we can assume that our Turing Machine has a unique end state. We just modify the Turing Machine to erase all the working tape whenever the algorithm accepts.*

Now we define the state transition function to be the one that takes two states, and outputs if the Turing machine can transition from the first state to the second in one step. Note we are talking about nondeterministic Turing Machines that could have multiple transitions, with deterministic Turing Machines as a special case.

Definition 3.22 (Turing Machine State Transition)**.** *For a Turing Machine A as in Definition 3.21, for any n, we define the state transition "matrix" as the function $M_n : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ such that:*

for any $s^0 = (q^0, h^0, w^0, m^0) \in \{0,1\}^S$ *and* $s^1 = (q^1, h^1, w^1, m^1) \in \{0,1\}^S$ *we have*

$$M_n(x, s^0, s^1) = 1 \iff \exists\, ((q_0, \sigma_h, \sigma_s), (q_1, d_h, d_w, \sigma_f)) = \lambda \in \Lambda : \tag{3}$$

$$q_0 = q^0 \wedge \tag{4}$$

$$q_1 = q^1 \wedge \tag{5}$$

$$\sigma_h = x_{h^0} \wedge \tag{6}$$

$$h^0 + d_h = h^1 \wedge \tag{7}$$

$$\sigma_s = m^0_{w^0} \wedge \tag{8}$$

$$\sigma_f = m^1_{w^0} \wedge \tag{9}$$

$$w^0 + d_w = w^1 \wedge \tag{10}$$

$$\forall j \in [S'] \setminus \{w^0\} : m^0_j = m^1_j. \tag{11}$$

Remark. *Condition 1 just specifies a state transition, λ. Conditions 2 and 3 verify the instruction state is updated according to λ. Conditions 4 and 5 verifies the input matches λ and the input head updates according to λ. Condition 6 and 7 verifies the working tape is updated according λ. Condition 8 verifies the working head updates according to λ. Condition 9 verifies nothing else on the working tape is changed.*

Remark. *We call this a transition matrix since after fixing x, M_x is just the adjacency matrix of the computation graph for A on input x, where $M_x(a,b) = M_{|x|}(x,a,b)$.*

Now we will show how to arithmetize the state transition function. In particular, we want a very low degree arithmetization we call a multilinear extension.

Definition 3.23 (Multilinear Extension)**.** *For a field $\mathbb{F}$, integer S and a function $\phi : \{0,1\}^S \to \mathbb{F}$, we define the multilinear extension of ϕ as the polynomial $\widehat{\phi} : \mathbb{F}^S \to \mathbb{F}$ that is degree 1 in any individual variable such that for all $a \in \{0,1\}^S$ we have $\phi(a) = \widehat{\phi}(a)$.*

We start by giving an arithmetization of the equality function, both as a warm up, and because it is an important component of our arithmetization.

Lemma 3.24 (Arithmetizing Equality)**.** *For any m, let $\mathrm{equ}' : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be such that $\mathrm{equ}'(a,b) = 1 \iff a = b$. Let $\mathrm{equ} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ be the multilinear extension of equ' for some finite field $\mathbb{F}$. ($\mathbb{F}$ and m will be clear from context).*

Then $\mathrm{equ}(x,y) = \prod_{i \in [m]} (x_i y_i + (1-x_i)(1-y_i))$, and $\mathrm{equ}(x,y)$ can be computed in time $\tilde{O}(\log(|\mathbb{F}|))m$ and space $O(\log(|\mathbb{F}|) + \log(m))$.

Proof. See this formula is multilinear since each term is multilinear in its x_i and y_i, and term is a function of different variables. Further, equ and equ′ agree on binary inputs, since each term is one if and only if $x_i = y_i$, and is 0 otherwise. Thus, equ is the unique multilinear extension of equ′. □

We use [Coo22a, Lemma 36]. We include the proof in Appendix A.

Theorem 3.25 (Arithmetization of State Transition)**.** *Suppose A is a space S nondeterministic algorithm with transition matrix $M_n : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ as described in Definition 3.22.*

Then we can compute the multilinear extension of M_n, denoted $\widehat{M_n} : \mathbb{F}^n \times \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$, in time $(n+S)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(S) + \log(n))\log(|\mathbb{F}|))$.

3.6 Standard Algebraic, Interactive Proof Tools

Now we will review many standard algebraic tools used in algebraic interactive proofs, like [GKR15; Coo22b; Lun+90; Sha92; She92]. Perhaps the most important tool is the original sum check from [Lun+90]. A proof of sum check can be found in [AB09]. For completeness, the proofs of the rest of these lemmas can be found in Appendix B.

Lemma 3.26 (Sum Check [Lun+90]). *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ has individual degree d. Then there is an S round interactive protocol with a verifier V that runs in time $Sd\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+d)\log(|\mathbb{F}|))$, and a prover P that runs in time $d2^S\tilde{O}(\log(|\mathbb{F}|))$ with $O(d2^S)$ oracle queries to f which takes as input $\alpha \in \mathbb{F}$ such that*

Completeness: *If $\sum_{w\in\{0,1\}^S} f(w) = \alpha$, then when V interacts with P, V outputs a uniform $w' \in \mathbb{F}^S$ and some $\alpha' \in \mathbb{F}$ such that $f(w') = \alpha'$.*

Soundness: *If $\sum_{w\in\{0,1\}^S} f(w) \neq \alpha$, then for any prover P' with probability at most $\frac{dS}{|\mathbb{F}|}$ will V output a uniform w' and some α' such that $f(w') = \alpha'$.*

There is an unlinearization protocol, like the one used by Shen [She92]. This can be proved using a sum check.

Lemma 3.27 (Unlinearization). *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ is a polynomial with individual degree d, and $\widehat{f}$ is the multilinear function consistent with f on binary inputs. Then there is an $S+1$ round interactive protocol with $O(dS\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $Sd\tilde{O}(\log(|\mathbb{F}|))$ and space $O((d+S)\log(|\mathbb{F}|))$, and a prover P that runs in time $d2^S\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(d2^S)$ oracle queries to f which takes as input a $w \in \mathbb{F}^S$ and $\alpha \in \mathbb{F}$ such that*

Completeness: *If $\widehat{f}(w) = \alpha$, then when V interacts with P, V outputs a $w' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ such that $f(w') = \alpha'$.*

Soundness: *If $\widehat{f}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{(d+1)S}{|\mathbb{F}|}$ will V output a w' and α' such that $f(w') = \alpha'$.*

There is a protocol to reduce a claim about a low degree polynomial at many points to a claim about a low degree polynomial at one point. This multi-point reduction is used in previous interactive proofs [GKR15; Coo22b] and similar query reductions have a long history in PCP literature [Aro+98; FL92; Din+99; Raz05; KR08].

Lemma 3.28 (Multi-Point Reduction). *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ has total degree d and m is some integer. Then there is a one round interactive protocol with $O(dm\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(S+d)m\tilde{O}(\log(|\mathbb{F}|))$ and space $O((md+S)\log(|\mathbb{F}|))$, and a prover P that runs in time $m^2dS\tilde{O}(\log(|\mathbb{F}|))$ with $O(dm)$ oracle queries to f. The protocol takes as input $(w_i \in \mathbb{F}^S)_{i\in[m]}$ and $(\alpha_i \in \mathbb{F})_{i\in[m]}$ and behaves such that*

Completeness: *If for each $i \in [m]$ we have $f(w_i) = \alpha_i$, then when V interacts with P, V outputs a $w' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ such that $f(w') = \alpha'$.*

Soundness: *If for any $i \in [m]$ we have $f(w_i) \neq \alpha_i$, then for any prover P' with probability at most $\frac{d(m-1)}{|\mathbb{F}|}$ will V output a w' and α' such that $f(w') = \alpha'$.*

A particular application of the multipoint reduction above is a product reduction.

Lemma 3.29 (Product Reduction). *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ has total degree d. Then there is a one round interactive protocol with $O(d\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(S+d)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+d)\log(|\mathbb{F}|))$, and a prover P that runs in time $Sd\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(d)$ oracle queries to the f. The protocol takes as input $u, v \in \mathbb{F}^S$ and $\alpha \in \mathbb{F}$ and acts such that*

Completeness: *If $f(u)f(v) = \alpha$, then when V interacts with P, V outputs a $w \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ such that $f(w) = \alpha'$.*

Soundness: *If $f(u)f(v) \neq \alpha$, then for any prover P' with probability at most $\frac{d}{|\mathbb{F}|}$ will V output a w and α' such that $f(w) = \alpha'$.*

4 Interactive Proof For Deterministic Algorithms

Internally, our proof will need interactive proofs for deterministic algorithms. So we start by proving a variation of the deterministic algorithms from [Coo22b; Tha20]. The algorithm is nearly the same, but we will need to verify the multilinear extension of the function the algorithm computes, instead of just a binary output.

The idea of the algorithm is that for a time T algorithm A, if on an input x algorithm A has computation graph G with adjacency matrix M, then for unique start state a and end state b, algorithm A accepts A if and only if $M^T_{a,b} = 1$. The by using a matrix square reduction repeatedly, this can be reduced to a statement about the value of $\widehat{M}$, the multilinear extension of M, at a random point. And $\widehat{M}$ can be calculated quickly using Theorem 3.25.

Our matrix square reduction is very similar to the matrix reduction in [Tha13], except generalized to the case where the matrix is also a multilinear extension of a third input.

Lemma 4.1 (Matrix Square To Matrix Reduction)**.** *Given a function $M : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \mathbb{F}$, denote for any $x \in \{0,1\}^n$ the matrix M_x such that $(M_x)_{u,v} = M(x,u,v)$. Then M^2_x is defined in the usual way: $(M^2_x)_{u,v} = \sum_{w\in\{0,1\}^S} M_x(u,w)M_x(w,v)$. Now define $M^2 : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \mathbb{F}$ by $M^2_x(u,v) = (M^2_x)_{u,v}$. Let $\widehat{M}$ be the multilinear extension of M and $\widehat{M^2}$ be the multilinear extension of M^2.*

Then there is an $S+n+2$ round interactive protocol with $O((S+n)\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(S+n)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+n)\log(|\mathbb{F}|))$, and a prover P that runs in time $2^{S+n}\tilde{O}(\log(|\mathbb{F}|))$ with $O(2^{S+n})$ oracle queries to $\widehat{M}$. The protocol takes as input $\alpha \in \mathbb{F}$, $u, v \in \mathbb{F}^S$, and $x \in \mathbb{F}^n$ and acts such that

Completeness: *If $\alpha = \widehat{M^2}(x,u,v)$, then when V interacts with P, V outputs a $u', v' \in \mathbb{F}^S$, $x' \in \mathbb{F}^n$, and $\alpha' \in \mathbb{F}$ such that $\alpha' = \widehat{M}(x',u',v')$.*

Soundness: *If $\alpha \neq \widehat{M^2}(x,u,v)$, then for any prover P' with probability at most $\frac{4S+3n}{|\mathbb{F}|}$ will V output a $u', v' \in \mathbb{F}^S$, $x' \in \mathbb{F}^n$, and $\alpha' \in \mathbb{F}$ such that $\alpha' = \widehat{M}(x',u',v')$.*

Proof. We first run the unlinearization Lemma 3.27 on the first input to reduce the claim about $\widehat{M^2}$ to a claim that

$$\alpha_0 = \sum_{w\in\{0,1\}^S} \widehat{M}(x',u,w)\widehat{M}(x',w,v).$$

Then we use sum check Lemma 3.26 to reduce this to a claim that

$$\alpha_1 = \widehat{M}(x',u',w)\widehat{M}(x',w,v').$$

Then we use a product reduction Lemma 3.29 to reduce this to a that

$$\alpha' = \widehat{M}(x',u',v').$$

The verifier runs unlinearization in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|))$ since

$$g(x^*) = \sum_{w\in\{0,1\}^S} \widehat{M}(x^*,u,w)\widehat{M}(x^*,w,v)$$

only has individual degree 2. The verifier runs the sum check in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|))$. The verifier runs the product reduction in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|))$. Thus the verifier runs in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|))$

The prover runs the unlinearization in time $2^n\tilde{O}(\log(|\mathbb{F}|))$ with $O(2^n)$ queries to

$$g(x^*) = \sum_{w\in\{0,1\}^S} \widehat{M}(x^*,u,w)\widehat{M}(x^*,w,v).$$

Each query to g takes time $2^S\tilde{O}(\log(|\mathbb{F}|))$ using 2^S queries to $\widehat{M}$. Thus the unlinearization takes prover time $2^{n+S}\tilde{O}(\log(|\mathbb{F}|))$ with 2^{n+S} oracle queries to $\widehat{M}$. The prover runs the sum check in time $2^S\tilde{O}(\log(|\mathbb{F}|))$ with $O(2^S)$ oracle queries to $\widehat{M}(y,u,w)\widehat{M}(y,w,v)$, which only requires $O(2^S)$ queries to $\widehat{M}$. The product reduction only takes prover time $S^2\tilde{O}(\log(|\mathbb{F}|))$ with $O(S)$ oracle queries to $\widehat{M}$. Thus the prover runs in time $2^{S+n}\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(2^{S+n})$ queries to $\widehat{M}$.

For completeness, suppose $\alpha = \widehat{M^2}(x,u,v)$. Then by completeness of unlinearization,

$$\alpha_0 = \sum_{w\in\{0,1\}^S} \widehat{M}(x',u,w)\widehat{M}(x',w,v).$$

By completeness of sum check,

$$\alpha_1 = \widehat{M}(x',u',w)\widehat{M}(x',w,v').$$

And by completeness of product reduction,

$$\alpha' = \widehat{M}(x',u',v').$$

For soundness, suppose $\alpha \neq \widehat{M^2}(x,u,v)$. By soundness of unlinearization, with probability at most $\frac{3n}{|\mathbb{F}|}$ will the verifier not reject and $\alpha_0 = \sum_{w\in\{0,1\}^S} \widehat{M}(x',u,w)\widehat{M}(x',w,v)$. If $\alpha_0 \neq \sum_{w\in\{0,1\}^S} \widehat{M}(x',u,w)\widehat{M}(x',w,v)$, then by soundness of sum check, with probability at most $\frac{2S}{|\mathbb{F}|}$ will $\alpha_1 = \widehat{M}(x',u',w)\widehat{M}(x',w,v')$. Then by soundness of product reduction, with probability at most $\frac{2S}{|\mathbb{F}|}$ will $\alpha' = \widehat{M}(x',u',v')$.

So by a union bound, the probability that $\alpha' = \widehat{M}(x',u',v')$ is at most $\frac{4S+3n}{|\mathbb{F}|}$. □

Now applying this square reduction $\log(T)$ times gives our interactive proof for the multilinear extension of a space efficient function.

Theorem 4.2 (Interactive Proof For Multilinear Extension of Bounded Space)**.** *For any function $D : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, for any $x \in \{0,1\}^n$, denote $D_x : \{0,1\}^m \to \{0,1\}$ by $D_x(y) = D(x,y)$. Let $\widehat{D_x}$ be the multilinear extension of D_x. If D is computed by a space S time T deterministic algorithm, then there is a $(m+S+2)\log(T)$ round interactive protocol with $O((m+S)\log(T)\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(n+(m+S)\log(T))\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(n)+m+S)\log(|\mathbb{F}|))$, and a prover P that runs in time $2^{2m+2S}\log(T)\tilde{O}(\log(|\mathbb{F}|))$ which takes as input an $x \in \{0,1\}^n$, $w \in \mathbb{F}^S$ and $\alpha \in \mathbb{F}$ such that*

Completeness: *If $\widehat{D_x}(w) = \alpha$, then when V interacts with P, V accepts.*

Soundness: *If $\widehat{D_x}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{(4S+3m)\log(T)}{|\mathbb{F}|}$ will V accept.*

Proof. The idea is to apply Lemma 4.1 $\log(T)$ times.

Let $M_n : \{0,1\}^n \times \{0,1\}^m \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be the state transition function of D such that $D(x,y,s^0,s^1) = 1$ if and only if D on input (x,y) and starting at state s^0 transitions to state s^1 after one step, as in Definition 3.22. For any $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, let $M_{x,y}$ be the matrix defined by $(M_{x,y})_{u,v} = M(x,y,u,v)$. Let a be the starting state of the algorithm, and b be the unique accept state.

Then observe that $(M^i_{x,y})_{u,v} = 1$ if and only if when D on input (x,y) starts at state u after i steps is at state v. Since D runs in time T, see that D accepts (x,y) if and only if $(M^T_{x,y})_{a,b} = 1$. That is, $D_x(y) = (M^T_{x,y})_{a,b}$. Now let $M^T_x : \{0,1\}^m \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be defined by $M^T_x(y,u,v) = (M^T_{x,y})_{u,v}$ and $\widehat{M^T_x} : \mathbb{F}^n \times \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$ be the multilinear extension of M^T_x. Then we have $\widehat{D}(x) = \widehat{M^T_x}(y,a,b)$.

We take T to be a power of 2 so that for some t we have $T = 2^t$. So the verifier runs Lemma 4.1 to reduce the claim that $\alpha = \widehat{M^T_x}(w,a,b)$ to the claim that $\alpha_1 = \widehat{M^{2^{t-1}}_x}(y_1,a_1,b_1)$. Then the verifier repeats this $\log(T)$ times to get the claim that $\alpha' = \widehat{M_x}(y',a',b') = \widehat{M_n}(x,y',a',b')$. Finally, the verifier runs Theorem 3.25 to calculate $\widehat{M_n}(x,y',a',b')$ and reject if it is not equal to α'.

This verifier takes time $(S+m)\log(T)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+m)\log(|\mathbb{F}|))$ to run Lemma 4.1 $\log(T)$ times. The verifier computes $\widehat{M_n}(x,y',a',b')$ in time $(S+n+m)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(S)+\log(n+m))\log(|\mathbb{F}|))$. Thus the verifier runs in time $(n+(S+m))\log(T)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+m+\log(n))\log(|\mathbb{F}|))$.

The prover first needs to calculate $M_{x,y}^{2^i}$ for every $i \in [\log(T)]$ and $y \in \{0,1\}^m$. First, we note that since D is deterministic, $M_{x,y}^{2^i}$ is sparse, as in each row has only one non zero entry. So we define $N_{x,y}^{2^i} : \{0,1\}^S \to \{0,1\}^S \cup \perp$ such that $N_{x,y}^{2^i}(u) = v$ if and only if $(M_{x,y}^{2^i})_{u,v} = 1$. Then the prover can calculate each $N_{x,y}^{2^i}$ in time $2^{m+S}\log(T)\tilde{O}(\log(|\mathbb{F}|))$.

Now see that given each of these, we can calculate any $\widehat{M_x^{2^i}}(y,u,v)$ in time $(m+S)2^{m+S}\tilde{O}(\log(|\mathbb{F}|))$, since there are only 2^S entries in any $M_{x,y}^{2^i}$. Specifically,

$$\widehat{M_x^{2^i}}(y,u,v) = \sum_{y^* \in \{0,1\}^m} \text{equ}(y,y^*) \sum_{z \in \{0,1\}^S} \text{equ}(u,z)\text{equ}(v, N_{x,y^*}^{2^i}(v)),$$

which can be evaluated straightforwardly in time $2^{m+S}\tilde{O}(\log(|\mathbb{F}|))$.

Now the prover in the Lemma 4.1 runs in time $m2^{m+S}\tilde{O}(\log(|\mathbb{F}|))$ with $O(2^{m+S})$ oracle queries to $\widehat{M^{2^i}}$. Thus each use of Lemma 4.1 can be run in time $2^{2m+2S}\tilde{O}(\log(|\mathbb{F}|))$. Thus the prover runs in time $2^{2m+2S}\log(T)\tilde{O}(\log(|\mathbb{F}|))$.

For completeness, suppose $\widehat{D_x}(w) = \alpha$. Then $\widehat{M_x^T}(w,a,b) = \alpha$, and by completeness of Lemma 4.1, we have $\alpha' = \widehat{M_n}(x,y',a',b')$. Thus the verifier accepts.

For soundness, suppose $\widehat{D_x}(w) \neq \alpha$. Then $\widehat{M_x^T}(w,a,b) \neq \alpha$, so by soundness of Lemma 4.1, with probability at most $\frac{4S+2m}{|\mathbb{F}|}$ will $\alpha_1 = \widehat{M_x^{2^{t-1}}}(y_1,a_1,b_1)$. Then by a union bound, with probability at most $\frac{(4S+2m)\log(T)}{|\mathbb{F}|}$ will $\alpha' = \widehat{M_n}(x,y',a',b')$. Thus with probability at most $\frac{(4S+3m)\log(T)}{|\mathbb{F}|}$ will the verifier accept. □

5 Interactive Proofs For Nondeterministic Algorithms

Now we give an interactive proof for nondeterministic algorithms because it is an interesting special case in its own right, it develops the tools needed for the more general alternating algorithm, and gives a good warm up for the general case.

But before we start, we quickly make a detour to explain that the matrix "multiplication" used here for nondeterministic algorithms is different than the one used for deterministic algorithms. For deterministic algorithms, we used standard multiplication and addition in some field. But for nondeterministic algorithms, we do binary matrix multiplication, with multiplication replaced with AND, and addition replaced with OR. To emphasize the difference, we use parentheses around the exponent to indicate we are performing binary matrix multiplication.

Definition 5.1 (Binary Matrix Multiplication and Existential Walks)**.** *Let $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be any function. Then by induction, define $M^{(1)} = M$ and for any i, define*

$$M^{(i+1)}(u,v) = \bigvee_w M^{(i)}(u,w)M(w,v).$$

See that if M is an adjacency matrix of a graph, then $M^{(i)}(s,t) = 1$ if and only if there is a path from s to t of length i.

Remark. *M^i is different from $M^{(i)}$ in that $M^{(i+1)}$ uses an OR function, whereas M^{i+1} uses a plus function. These are equivalent for the adjacency matrix of a deterministic algorithm, but crucially differ for a nondeterministic algorithm.*

We emphasize that these binary matrix multiplications algebraically act very similarly to integer matrix multiplication. Specifically, $M^{(T)}$ can still be calculated with $\log(T)$ repeated binary squaring.

Now our goal is to replace the matrix sum check used for deterministic algorithms, with a new efficient reduction for nondeterministic algorithms. It is not clear how to do this in general, so we use a Razborov-Smolensky style low degree approximation, and give a reduction for that instead.

5.1 Extended Product Reduction

The main tool for this new reduction is this extended product reduction. This reduces a statement about the *multilinear extension* of a large product of terms to a statement about the *multilinear extension* of one term. This product reduction could be used to give a square reduction for nondeterministic algorithms directly, but is much more efficient if the number of multiplications is smaller. This is why we use Razborov-Smolensky.

The idea is to just apply many unlinearizations and product reductions, to one variable the product ranges over at a time. First we prove it for one variable.

Lemma 5.2 (Single Variable Extended Product Reduction)**.** *Suppose $\widehat{f} : \mathbb{F} \times \mathbb{F}^S \to \mathbb{F}$ is multilinear. Suppose $g : \mathbb{F}^S \to \mathbb{F}$ is the function $g(v) = \widehat{f}(0, v)\widehat{f}(1, v)$ and $\widehat{g}$ is the multilinear extension of g.*

Then there is an $S + 1$ round interactive protocol with $O(S \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S \log(|\mathbb{F}|))$, and a prover P that runs in time $2^S\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(2^S)$ oracle queries to f. The protocol takes as input $w \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ and acts such that

Completeness: *If $\widehat{g}(w) = \alpha$, then when V interacts with P, V outputs a $u' \in \mathbb{F}$, $v' \in \mathbb{F}^S$, and $\alpha' \in \mathbb{F}$ such that $\widehat{f}(u', v') = \alpha'$.*

Soundness: *If $\widehat{g}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{3S+1}{|\mathbb{F}|}$ will V output a $u' \in \mathbb{F}$, $v' \in \mathbb{F}^S$, and $\alpha' \in \mathbb{F}$ such that $\widehat{f}(u', v') = \alpha'$.*

Proof. The protocol first runs a unlinearization Lemma 3.27 to reduce to the claim that

$$\alpha_1 = g(v') = \widehat{f}(0, v')\widehat{f}(1, v').$$

Then we run our product reduction Lemma 3.29 on the first variable to reduce this to the claim that

$$\alpha' = \widehat{f}(u', v').$$

The verifier for unlinearization runs in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S \log(|\mathbb{F}|))$. The verifier for product reduction runs in time $\tilde{O}(\log(|\mathbb{F}|))$ and space $O(\log(|\mathbb{F}|))$. So the verifier runs in time $S\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S \log(|\mathbb{F}|))$.

The prover for unlinearization runs in time $2^S\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(2^S)$ oracle queries to g or equivalently $\widehat{f}$. The prover for the product reduction runs in time $\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(1)$ oracle queries to the f. So the prover runs in time $2^S\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(2^S)$ oracle queries to $\widehat{f}$.

Completeness holds due to completeness of Lemma 3.27 and Lemma 3.29.

For soundness, suppose $\widehat{g}(w) \neq \alpha$. Then by soundness of unlinearization, with probability at most $\frac{3S}{|\mathbb{F}|}$ will $\alpha_1 = \widehat{f}(0, v')\widehat{f}(1, v')$. If $\alpha_1 \neq \widehat{f}(0, v')\widehat{f}(1, v')$, by soundness of the product reduction, with probability at most $\frac{1}{|\mathbb{F}|}$ will $\alpha' = \widehat{f}(u', v')$. So by a union bound, with probability at most $\frac{3S+1}{|\mathbb{F}|}$ will $\alpha' = \widehat{f}(u', v')$. □

And now for several at a time.

Lemma 5.3 (Extended Product Reduction)**.** *Suppose $\widehat{f} : \mathbb{F}^\ell \times \mathbb{F}^S \to \mathbb{F}$ is multilinear. Let $g : \{0,1\}^S \to \mathbb{F}$ be defined by $g(v) = \prod_{u \in \{0,1\}^\ell} \widehat{f}(u, v)$ and let $\widehat{g}$ be the multilinear extension of g.*

Then there is an $\ell(S + 1)$ round interactive protocol with $O(\ell S \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\ell + S) \log(|\mathbb{F}|))$, and a prover P that runs in time $2^{\ell+S}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $w \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{g}(w) = \alpha$, then when V interacts with P, V outputs a $u' \in \mathbb{F}^\ell$, $v' \in \mathbb{F}^S$, and $\alpha' \in \mathbb{F}$ such that $\widehat{f}(u', v') = \alpha'$.*

Soundness: *If $\widehat{g}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{l(3S+1)}{|\mathbb{F}|}$ will V output a $u' \in \mathbb{F}^\ell$, $v' \in \mathbb{F}^S$, and $\alpha' \in \mathbb{F}$ such that $\widehat{f}(u', v') = \alpha'$.*

Proof. The idea is to apply Lemma 5.2 ℓ times.

Define $f_1 : \mathbb{F} \times \mathbb{F}^S \to \mathbb{F}$ by $f_1(x, v) = \prod_{u \in \{0,1\}^{\ell-1}} \widehat{f}((x, u), v)$ and let $\widehat{f_1}$ be the multilinear extension of f_1. Let $g_1 : \mathbb{F}^S \to \mathbb{F}$ be defined by $g_1(v) = \widehat{f}(0, v)\widehat{f}(1, v)$ and $\widehat{g_1}$ be the multilinear extension of g_1.

Now see that $\widehat{g_1}$ is multilinear and agrees with $\widehat{g}$ on Boolean assignments. Thus $\widehat{g_1} = \widehat{g}$. So $\widehat{g}(w) = \alpha \equiv \widehat{g_1}(w)$. So our extended product reduction reduces to the claim that $\alpha_1 = \widehat{f_1}(u'_1, v_1)$.

We similarly define f_2 as $f_2(x, v) = \prod_{u \in \{0,1\}^{\ell-2}} \widehat{f}((u'_1, x, u), v)$, $\widehat{f_2}$ as its multilinear extension, g_2 as $g_2(v) = \widehat{f_2}(0, v)\widehat{f_2}(1, v)$ and $\widehat{g_2}$ as its multilinear extension. Then we note that $\widehat{g_2}(v) = \widehat{f_1}(u'_1, v)$ since both are multilinear and agree on Boolean inputs. So $\widehat{g_2}(v_1) = \alpha_1 \equiv \alpha_1 = \widehat{f_1}(u'_1, v_1)$. Then we run Lemma 5.2 again.

After running the reduction ℓ times, we get the claim that $\alpha' = \widehat{f}(u', v')$.

The verifier time and space is just the time and space to run Lemma 5.2 ℓ times, plus the space to hold the resulting u' and v'. This is time $\ell S\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\ell + S)\log(|\mathbb{F}|))$.

The prover needs to provide oracle access to each $\widehat{f_i}$. An oracle call to $\widehat{f_i}$ just uses $2^{\ell-i}$ oracle calls to $\widehat{f}$ and takes time $2^{\ell-i}\tilde{O}(\log(|\mathbb{F}|))$. So for the ith invocation of Lemma 5.2, the prover runs in time $(2^S\tilde{O}(\log(|\mathbb{F}|))$ plus time $2^S 2^{\ell-i}\tilde{O}(\log(|\mathbb{F}|))$ and $O(2^S 2^{\ell-i})$ oracle calls to $\widehat{f}$ to make the 2^S oracle calls. In total, this is time $2^{S+\ell}\tilde{O}(\log(|\mathbb{F}|))$ with $O(2^{S+\ell})$ oracle calls to f.

Completeness holds by completeness of Lemma 5.2. For soundness, this protocol only fails if for some i we have $\alpha_i = \widehat{f_i}$. By soundness of Lemma 5.2 and a union bound, this only happens with probability at most $\frac{l(2S+1)}{|\mathbb{F}|}$. □

5.2 Low degree Approximations

To make this proof work, we need to be able to sample D_r which works with high probability that can be calculated efficiently. Luckily, our D_r just needs to sample from a bunch of ϵ-biased sets. Construction of efficient ϵ-biased sets is well researched and very efficient constructions are known [NN93; TS17] and is equivalent to constructing good, linear codes. Then to sample the ϵ bias sets, we can use any efficient hitting sampler. We use random walks on an expander graph.

We use the third construction in [Alo+90] as our ϵ biased sets. This is the same ϵ biased set used in [GR20], except that we need to sample them more efficiently. A description and proof is included in Appendix C.

Lemma 5.4 (ϵ-Biased Set)**.** *For any S, there is an $m = O(S)$ and a function $D' : \{0,1\}^m \times \{0,1\}^S \to \{0,1\}$ such that for any $X \subseteq \{0,1\}^S \setminus \emptyset$*

$$\Pr_{r \in \{0,1\}^m}\left[\sum_{x \in X} D'(r, x) = 1 \pmod 2\right] \geq \frac{1}{4}$$

such that D' runs in $\mathrm{poly}(S)$ time and $O(S)$ space.

Now we have a D' which with constant probability correctly converts an OR to a parity. Now we need to sample enough of these so that with constant probability, we convert 2^S ORs into parities. If we take $O(S)$ independent samples of D', then with probability less than 2^{-2S} will any of these ORs fail to be converted into a parity, so by a union bound with probability at most 2^{-S} will any of them fail to be converted into parity. Of course, we can not afford to take $O(S)$ samples of a string of length $O(S)$. So we take correlated samples using random walks on an expander graph.

The following is a direct consequence of Lemma 3.19 and Lemma 3.20. A proof is in Appendix C.

Lemma 5.5 (Efficient Hitting Sampler)**.** *For any m, and any $\epsilon > 0$, there is an $L = 2^\ell = O(\log(1/\epsilon))$, an $R = O(m + L)$ and a function $W : \{0,1\}^R \times \{0,1\}^\ell \to \{0,1\}^m$ such that, if for any $D^* : \{0,1\}^m \to \{0,1\}$ we have that $\Pr_r[D^*(r) = 1] \geq \frac{1}{4}$, then*

$$\Pr_{r \in \{0,1\}^R}[\forall a \in \{0,1\}^\ell : D^*(W(r,a)) = 0] \leq \epsilon.$$

Further, W is computable in time $\text{poly}(R)$ and space $O(R)$.

Now we construct our good ϵ biased set sampler.

Theorem 5.6 (Efficient Epsilon Biased Set Sampler)**.** *For any $n = 2^S$ and $\epsilon > 0$, for some $L = 2^\ell = O(\log(1/\epsilon))$ and $R = O(S + L)$, there is a function $D : \{0,1\}^R \times \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$. For $r \in \{0,1\}^R$, define $D_r : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$.*

D is such that for any $A \subseteq \{0,1\}^S$ with $A \neq \emptyset$, we have

$$\Pr_r[\forall i \in \{0,1\}^\ell : \sum_{j \in A} D_r(i,j) = 0 \pmod 2] \leq \epsilon.$$

Further, D is computable by a space $O(R)$ time $\text{poly}(R)$ algorithm.

Proof. Our function D is just our hitting sampler Lemma 5.5 (which is just random walks on an expander) composed with an ϵ bias set Lemma 5.4.

Specifically, let $D' : \{0,1\}^m \times \{0,1\}^S \to \{0,1\}$ be the small bias function from Lemma 5.4. See that $m = O(S)$. Let $D^* : \{0,1\}^m \to \{0,1\}$ be the function outputting if D' has parity one on $A \subseteq \{0,1\}^S$. That is $D^*(i) = 1$ if and only if $\sum_{j \in A} D'(i,j) = 1 \mod 2$. Then by Lemma 5.4, $\Pr_r[D^*(i) = 1] \geq \frac{1}{4}$.

Let $W : \{0,1\}^R \times \{0,1\}^\ell \to \{0,1\}^m$ be from Lemma 5.5. See that $L = O(\log(1/\epsilon))$ and $R = O(m + L) = O(S + \log(1/\epsilon))$. Let $D_r(i,j) = D'(W(r,i),j)$. Then by Lemma 5.5,

$$\begin{aligned}\epsilon &\geq \Pr_{r \in \{0,1\}^R}[\forall a \in \{0,1\}^\ell : D^*(W(r,a)) = 0] \\ &= \Pr_{r \in \{0,1\}^R}[\forall a \in \{0,1\}^\ell : \sum_{j \in A} D'(W(r,a),j) = 0 \pmod 2].\end{aligned}$$

□

In particular, we generally use the following result to show our D is often good for many, many ORs at once with a modest amount of randomness.

Lemma 5.7 (Sampling a Good D for Many ORs)**.** *Suppose for $n = 2^S$ and $m = 2^{S'}$, for each $i \in [m]$ there is an $f_i \in \{0,1\}$ and $u^i \in \{0,1\}^i$ such that*

$$f_i = \bigvee_{j \in [n]} u^i_j.$$

Then for any ϵ, for $R = O(S + S' + \log(1/\epsilon))$, $L = 2^\ell = O(S' + \log(1/\epsilon))$, the D from Theorem 5.6, for each $i \in [m]$ define

$$F^r_i = 1 + \prod_{k \in \{0,1\}^\ell} (1 + \sum_{j \in [n]} D_r(k,j) u^i_j) \mod 2.$$

If $\forall i \in [m] : F^r_i = f_i$, then we say D_r is good for each f.

Then

$$\Pr_r[D_r \text{ is not good for } f] < \epsilon.$$

Proof. The idea is that $F_i^r = f_i$ is exactly equivalent to some parities of D_r being one. So by using Theorem 5.6 and a union bound, we can get calculate all of these correctly with high probability.

First, see that for every $i \in [m]$ and $r \in \{0,1\}^R$ we have that $f_i = 0 \implies F_i^r = 0$. If $f_i = 0$, then for each $j \in [n]$, $u_j^i = 0$. So for every k, $\sum_{j\in[n]} D_r(k,j)u_j^i = 0$. Thus $F_i^r = 0$. So if $f_i = 0$, we have

$$\Pr_r[F_i^r \neq f_i] = 0 < \epsilon'.$$

So suppose $f_i \neq 0$. Then let A be the set of $j \in [n] = \{0,1\}^S$ such that $u_j^i \neq 0$. Since $f_i = 1$, we know $A \neq \emptyset$. Set $\epsilon' = \frac{\epsilon}{2^{S'}}$. Then by Theorem 5.6, for $L = 2^\ell = O(S' + \log(1/\epsilon))$ and $R = O(S + S' + \log(1/\epsilon))$ we have

$$\begin{aligned}
\epsilon' &\geq \Pr_r\left[\forall k \in \{0,1\}^\ell : \sum_{j\in A} D_r(k,j) = 0 \pmod 2\right] \\
&= \Pr_r\left[\forall k \in \{0,1\}^\ell : \sum_{j\in[n]} D_r(k,j)u_j^i = 0 \pmod 2\right] \\
&= \Pr_r\left[\prod_{k\in\{0,1\}^\ell} (1 + \sum_{j\in[n]} D_r(k,j)u_j^i) = 1 \pmod 2\right] \\
&= \Pr_r\left[1 + \prod_{k\in\{0,1\}^\ell} (1 + \sum_{j\in[n]} D_r(k,j)u_j^i) = 0 \pmod 2\right] \\
&= \Pr_r[F_i^r \neq f_i].
\end{aligned}$$

Thus for any i, we have $\Pr_r[F_i^r \neq f_i] < \epsilon'$. And since $\epsilon' = \frac{\epsilon}{2^{S'}}$, by a union bound over all $m = 2^{S'}$ choices of i, the probability any of them are not equal is at most ϵ. □

Remark. *One can use DeMorgans to convert any AND to an OR. So*

$$\bigvee_{j\in[n]} u_j^i = \neg \bigwedge_{j\in[n]} \neg u_j^i.$$

5.3 Interactive Proofs For Nondeterministic Algorithms

Now we are ready to give our proof for nondeterministic algorithms. We do this by defining our interactive proof we wish to run, assuming we got a good D_r. This is a square reduction, assuming we can use the Razborov-Smolensky formula to describe $M^{(2)}$. We call the Razborov-Smolensky style polynomial given by our pseudorandomness D as "M relative to D". We say that our D_r is good if $M^{(T)}$ relative to D is $M^{(T)}$. If D_r is good, we are done. Otherwise, D_r is bad, and makes some mistake first. Then we give an interactive proof to show where it is bad.

First, we formally define M relative to D.

Definition 5.8 (M Relative to D)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, and $D : \{0,1\}^\ell \times \{0,1\}^S \to \mathbb{F}$, we define M relative to D as the functions, for $k = 1$, $M_D^{(1)} = M_D = M$, and for any $k > 1$ the function $M_D^{(2^k)} : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ is*

$$M_D^{(2^k)}(u,v) = 1 + \prod_{j\in\{0,1\}^\ell} (1 + \sum_{w\in\{0,1\}^S} D(j,w)M_D^{(2^{k-1})}(u,w)M_D^{(2^{k-1})}(w,v)).$$

Now we give our square reduction for $M^{(2)}$, relative to D.

Lemma 5.9 (Square Reduction For M Relative To D)**.** *For some $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, let $\widehat{M}$ be the multilinear extension of M and $\widehat{M^{(2)}}$ be the multilinear extension of $M^{(2)}$. For some $D : \{0,1\}^\ell \times \{0,1\}^S \to \mathbb{F}$ let $\widehat{M_D^{(2)}}$ be the multilinear extension of M relative to D given by Definition 5.8.*

Then there is an $O(\ell S)$ round interactive protocol with $O(\ell S \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S \tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+\ell)\log(|\mathbb{F}|))$, and a prover P (with access to the truth table of M and D) that runs in time $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{M_D^{(2)}}(u,v) = \alpha$, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u',v') = \alpha'$ and $D(j',w') = \beta'$.*

Soundness: *If $\widehat{M_D^{(2)}}(u,v) \neq \alpha$, then for any prover P' with probability at most $\frac{(\ell+1)(6S+1)}{|\mathbb{F}|}$ will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u',v') = \alpha'$ and $D(j',w') = \beta'$.*

Proof. The protocol simply runs our extended product reduction Lemma 5.3, a sum check Lemma 3.26, and finally a product reduction Lemma 3.29.

For any $u, v \in \mathbb{F}^S$, we define $\tilde{F} : \mathbb{F}^\ell \times \mathbb{F}^S \times \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$ by

$$\tilde{F}(j,u,v,w) = \widehat{D}(j,w)\widehat{M}(u,w)\widehat{M}(w,v).$$

See that $\tilde{F}$ is multilinear in j, u, v and has individual degree 3 in w.

Let $\widehat{G} : \mathbb{F}^\ell \times \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$ be the function

$$\widehat{G}(j,u,v) = 1 + \sum_{w \in \{0,1\}^S} \tilde{F}(j,u,v,w).$$

See that $\widehat{G}$ is multilinear since $\tilde{F}$ is in its first three arguments.

Let $H : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be defined by

$$H(u,v) = \prod_{j \in \{0,1\}^\ell} \widehat{G}(j,u,v).$$

Let $\widehat{H}$ be the multilinear extension of H.

By our definition of $M_D^{(2)}$, we have for any $u, v \in \{0,1\}^S$ that

$$\begin{aligned}
M_D^{(2)}(u,v) =& 1 + \prod_{j \in \{0,1\}^\ell} (1 + \sum_{w \in \{0,1\}^S} D(j,w)M(u,w)M(w,v)) \\
1 + M_D^{(2)}(u,v) =& \prod_{j \in \{0,1\}^\ell} (1 + \sum_{w \in \{0,1\}^S} \tilde{F}(j,u,v,w)) \\
=& \prod_{j \in \{0,1\}^\ell} \widehat{G}(j,u,v) \\
=& H(u,v).
\end{aligned}$$

Thus since both $1 + \widehat{M_D^{(2)}}$ and $\widehat{H}$ are multilinear and agree on binary inputs, they are equal. So a claim that $\widehat{M_D^{(2)}}(u,v) = \alpha$ is equivalent to a claim that

$$\alpha + 1 = \widehat{H}(u,v).$$

So by applying the extended product reduction, Lemma 5.3, we reduce to the claim that $\alpha + 1 = H(u, v)$ to the claim that for some $\alpha_1 \in \mathbb{F}$, $j' \in \mathbb{F}^\ell$, and $u^*, v^* \in \mathbb{F}^S$ we have that

$$\begin{aligned}\alpha_1 =& \widehat{G}(j', u^*, v^*)\\ =& 1 + \sum_{w \in \{0,1\}^S} \tilde{F}(j', u^*, v^*, w)\\ \alpha_1 + 1 =& \sum_{w \in \{0,1\}^S} \tilde{F}(j', u^*, v^*, w).\end{aligned}$$

Then applying sum check, Lemma 3.26, we reduce to the claim that for some $\alpha_2 \in \mathbb{F}$ and $w' \in \mathbb{F}^S$ that

$$\begin{aligned}\alpha_2 =& \tilde{F}(j', u^*, v^*, w')\\ =& \widehat{D}(j', w')\widehat{M}(u^*, w')\widehat{M}(w', v^*).\end{aligned}$$

Next the prover provides $\beta' \in \mathbb{F}$ with the claim that $\beta' = \widehat{D}(j', w')$ and $\alpha_3 \in \mathbb{F}$ with the claim that $\alpha_3 = \widehat{M}(u^*, w')\widehat{M}(w', v^*)$. If $\alpha_3\beta' \neq \alpha_2$, the verifier rejects.

Now we apply the product reduction Lemma 3.29 to get the claim that for some $\alpha' \in \mathbb{F}$ and $u', v' \in \mathbb{F}^S$ we have

$$\alpha' = \widehat{M}(u', v').$$

The verifier time is just the sum of each of Lemma 5.3, Lemma 3.26, and Lemma 3.29. This is time

$$(\ell S + S + S)\tilde{O}(\log(|\mathbb{F}|)) = \ell S\tilde{O}(\log(|\mathbb{F}|)).$$

The verifier space is just the max space of any of these, which is

$$O((S + \ell)\log(|\mathbb{F}|)).$$

For the prover time, we just add the prover times. But we need to be able to compute the oracles for each prover as well. Notably an oracle to $\tilde{F}, \widehat{G}, \widehat{H}$, or $\widehat{M}$ can be calculated in time $2^{\ell+3S}\tilde{O}(\log(|\mathbb{F}|))$. So the prover can run in time[6]

$$2^{\ell+3S}2^{\ell+3S}\tilde{O}(\log(|\mathbb{F}|)) = 2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|)).$$

Completeness holds by completeness of each of each of Lemma 5.3, Lemma 3.26, and Lemma 3.29. By a union bound, the soundness is the sum of their soundness, which is

$$\frac{l(6S+1) + 3S + S}{|\mathbb{F}|} = \frac{(\ell+1)(6S+1)}{|\mathbb{F}|}.$$

□

Now we show our repeated square reduction for M relative to D. If D is good pseudorandomness, this gives our interactive protocol for nondeterministic algorithms.

Lemma 5.10 (Repeated Square Reduction For M relative to D)**.** *For some $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, let $\widehat{M}$ be the multilinear extension of M and $\widehat{M^{(2)}}$ be the multilinear extension of $M^{(2)}$. For some $D : \{0,1\}^\ell \times \{0,1\}^S \to \mathbb{F}$ and $T = 2^t$ let $\widehat{M_D^{(T)}}$ be the multilinear extension of M relative to D given by Definition 5.8.*

Then there is an $O(\ell S \log(T))$ round interactive protocol with $O(\ell S \log(T)\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S \log(T)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+\ell)\log(|\mathbb{F}|))$, and a prover P (with access to the truth table of M and D) that runs in time $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

[6] We can do better by calculating the prover messages in the subroutines directly. This still requires computing $\widehat{F}$, which takes time $2^{\ell+3S}\tilde{O}(\log(|\mathbb{F}|))$.

Completeness: *If* $\widehat{M_D^{(T)}}(u,v) = \alpha$*, then when* V *interacts with* P*,* V *outputs a* $u', v', w' \in \mathbb{F}^S$*,* $j' \in \mathbb{F}^\ell$*, and* $\alpha', \beta' \in \mathbb{F}$ *such that* $\widehat{M}(u', v') = \alpha'$ *and* $D(j', w') = \beta'$*.*

Soundness: *If* $\widehat{M_D^{(T)}}(u,v) \neq \alpha$*, then for any prover* P' *with probability at most* $\frac{\log(T)(\ell+2)(6S+2)}{|\mathbb{F}|}$ *will* V *output a* $u', v', w' \in \mathbb{F}^S$*,* $j' \in \mathbb{F}^\ell$*, and* $\alpha', \beta' \in \mathbb{F}$ *such that* $\widehat{M}(u', v') = \alpha'$ *and* $D(j', w') = \beta'$*.*

Proof. This is found by applying our square reduction Lemma 5.9 for $\log(T)$ times and then using a union bound. The main caveat is that each time we do this, we get a different claim about the value of D at a different place. But we can reduce this to one location using our multi-point reduction Lemma 3.28. To keep the space low, the verifier needs to use this after each application of Lemma 5.9.

Completeness holds by completeness of Lemma 3.28 and Lemma 5.9.

So the soundness for a single round of Lemma 5.9 and Lemma 3.28 is just

$$\frac{(\ell+1)(6S+1)+S+\ell}{|\mathbb{F}|} \leq \frac{(\ell+2)(6S+2)}{|\mathbb{F}|}.$$

Similarly the verifier time is just

$$(\ell S + \ell + S)\tilde{O}(\log(|\mathbb{F}|)) = \ell S \tilde{O}(\log(|\mathbb{F}|)).$$

Then to get the total soundness and verifier time, we just multiply these by $\log(T)$.

The verifier space is just the max space for any of these steps, which is just

$$O((\ell + S + \ell + S)\log(|\mathbb{F}|)) = O((\ell + S)\log(|\mathbb{F}|)).$$

The verifier does need to use $\log(\log(T))$ bits to store the current step, but using the trivial bound that $T < 2^S$, we only need to use S bits for that.

To run the prover, the prover needs to provide the truth tables of $M_D^{(2^k)}$ for each $k \leq \log(T)$ and to D. This takes time

$$O(\log(T)2^{2S+\ell+S} + S2^{2S} + (\ell+S)2^{\ell+S}) = 2^{O(\ell+S)}.$$

Then the rest of the prover time is the sum of the prover times for the subroutines, which is just

$$\log(T)2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|)) = 2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|)).$$

□

Unfortunately, D is not always good. So we formally define when D is or is not good for M, then show how to prove it is not good.

Definition 5.11 (D is Good for M up to time T)**.** *For any* $T = 2^t$*, we say that* D *is good for* M *up to time* T *if for all* $k \in [t]$ *we have*

$$M^{(2^k)} = M_D^{(2^k)}.$$

We note that by Theorem 5.6 and Lemma 5.7, we can sample a good D with high probability efficiently, and that D can be calculated efficiently. Then when D is good, by Lemma 5.10, we have an efficient interactive protocol for $M^{(T)}$. All that is left is to show that when D is bad, we can prove D is bad with perfect completeness.

Lemma 5.12 (Proving D is Bad for M with Time)**.** *For some* $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$*, and integer* $T = 2^t$*, let* $\widehat{M}$ *be the multilinear extension of* M *and for any* k *let* $\widehat{M^{(2^k)}}$ *be the multilinear extension of* $M^{(2^k)}$*. Let* ℓ *be an integer and* $D : \mathbb{F}^\ell \times \mathbb{F}^S \to \mathbb{F}$ *a multilinear function.*

Then there is an $O(\ell S \log(T))$ *round interactive protocol with* $O(\ell S \log(T)\log(|\mathbb{F}|))$ *bits of communication, a verifier* V *that runs in time* $\ell S \log(T)\tilde{O}(\log(|\mathbb{F}|))$ *and space* $O((S+\ell)\log(|\mathbb{F}|))$*, and a prover* P *(with access to the truth table of* M *and* D*) that runs in time* $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ *which takes as input* $u, v \in \mathbb{F}^S$*, and* $\alpha \in \mathbb{F}$ *such that*

Completeness: *If D is not good for M up to T, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $D(j', w') = \beta'$.*

Soundness: *If D is good for M up to time T, then when V interacts with P, with probability at most $\frac{\log(T)(\ell+3)(6S+2)}{|\mathbb{F}|}$ will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $D(j', w') = \beta'$.*

Proof. The idea is the same as [GR20]. If D is not good, we ask the prover for the smallest power it is not good, and then run the interactive proof on that claim. The verifier can then run the interactive proof on these locations since D is good for all smaller powers.

If D is not good for M up to T, there must be a $T' = 2^{t'} \geq 1$ that D is good for M up to T', but D is not good for M up to time $2T' \leq T$. If M is not good for $2T'$, then that means there is some $u, v \in \{0,1\}^S$ such that

$$\begin{aligned} M_D^{(2T')}(u,v) \neq & M^{(2T')}(u,v) \\ = & \bigvee_{w \in \{0,1\}^S} M^{(T')}(u,w) M^{(T')}(w,v). \end{aligned}$$

Note that if $M^{(2T')}(u,v) = 0$, then for each $w \in \{0,1\}^S$ that $M^{(T')}(u,w)M^{(T')}(w,v)) = 0$. Since D is good for M up to time T', this would mean that

$$\begin{aligned} M_D^{(2T')}(u,v) =& 1 + \prod_{j \in \{0,1\}^\ell} (1 + \sum_{w \in \{0,1\}^S} D(j,w) M_D^{(T')}(u,w) M_D^{(T')}(w,v)) \\ =& 1 + \prod_{j \in \{0,1\}^\ell} (1 + \sum_{w \in \{0,1\}^S} D(j,w) M^{(T')}(u,w) M^{(T')}(w,v)) \\ =& 1 + \prod_{j \in \{0,1\}^\ell} (1+0) \\ =& 0 \\ =& M^{(2T')}(u,v). \end{aligned}$$

But we chose u and v such that $M_D^{(2T')}(u,v) \neq M^{(2T')}(u,v)$. So $M^{(2T')}(u,v) = 1$.

Thus it must be the case that $M_D^{(2T')}(u,v) = 0$, but $M^{(2T')}(u,v) = 1$. More specifically, it must be the case that for some $w \in \{0,1\}^S$ we have $M^{(T')}(u,w)M^{(T')}(w,v) = 1$. Thus our prover just needs to provide $u, v, w \in \{0,1\}^S$ and then:

- Run the square reduction Lemma 5.9 to reduce the claim that $M_D^{(2T')}(u,v) = 0$ to the claim that for some $u_0, v_0, w_0 \in \mathbb{F}^S$, $j_0 \in \mathbb{F}^\ell$, and $\alpha_0, \beta_0 \in \mathbb{F}$ that $\widehat{M_D^{T'}}(u_0, v_0) = \alpha_0$ and $D(j_0, w_0) = \beta_0$. Since D is good for M up to time D, see that $\widehat{M_D^{T'}}(u_0, v_0) = \widehat{M^{T'}}(u_0, v_0)$.
- Run the multi-point reduction Lemma 3.28 to reduce to the claim that $\widehat{M^{T'}}(u_0, v_0) = \alpha_0$, $M^{(T')}(u,w) = 1$ and $M^{(T')}(w,v) = 1$ to the claim that for some $\alpha_1 \in \mathbb{F}$ and some $u_1, v_1 \in \mathbb{F}^S$ that $\widehat{M^{T'}}(u_0, v_0) = \alpha_1$.
- Run the repeated square reduction Lemma 5.10 to reduce the claim that $\widehat{M^{T'}}(u_0, v_0) = \alpha_1$ to the claim that for some $u', v', w_2 \in \mathbb{F}^S$, $j_2 \in \mathbb{F}^\ell$, and $\alpha', \beta_2 \in \mathbb{F}$ we have that $\widehat{M}(u', v') = \alpha'$ and $D(j_2, w_2) = \beta_2$.
- Finally, run the multi-point reduction Lemma 3.28 to reduce the claims that $D(j_0, w_0) = \beta_0$ and $D(j_2, w_2) = \beta_2$ to the claim that for some $j' \in \mathbb{F}^\ell$, $w' \in \mathbb{F}^S$, and $\beta' \in \mathbb{F}$ that $D(j', w') = \beta'$.

,

The verifier time is just the sum of all the times of these protocols, which is just $\ell S \log(T)\tilde{O}(\log(|\mathbb{F}|))$ and the space is the maximum of any of these sub protocols, which is just $O((S+\ell)\log(|\mathbb{F}|))$.

To run the prover, the prover needs to provide the truth tables of $M_D^{(2^k)}$ and $M^{(2^k)}$ for each $k \leq \log(T)$ and the truth table of D. The prover has to compute $M^{(2^k)}$ to find the u, v such that $M^{(2^k)}(u,v) \neq M_D^{(2^k)}(u,v)$ This takes time

$$O(\log(T)(2^{2S+\ell+S} + 2^{2S}) + S2^{2S} + (\ell+S)2^{\ell+S}) = 2^{O(\ell+S)}.$$

Then the rest of the prover time is the sum of the prover times for the subroutines, which is $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$.

The completeness holds by completeness of the sub protocols. The soundness is the sum of the soundness of the parts, so the soundness is

$$\frac{(\ell+1)(6S+1) + 4S + \log(T')(\ell+2)(6S+2) + (\ell+S)}{|\mathbb{F}|} = \frac{\log(T)(\ell+3)(6S+2)}{|\mathbb{F}|}.$$

□

Finally, we show our final interactive proof for nondeterministic algorithms. We note here that we show more exactly that the polylogarithmic overhead in our verifier time is $\tilde{O}(\log(S)^2)$. This is worse than the polylogarithmic overhead for deterministic algorithms given in [Coo22b], which was $\tilde{O}(\log(S))$. This is because our extended product reduction is slower than a sum check, but only a $\log(S)$ factor slower.

Theorem 5.13 (Interactive Proof For Nondeterministic Time and Space)**.** *For any T, S, d, and ϵ constructible in time T and space S:*

$$\mathbf{NTISP}[T,S] \subseteq \mathbf{ITIME}^1\left[(n + S\log(T))\tilde{O}(\log(S)^2), 2^{O(S)}\right].$$

Further, the verifier runs in space $O(S\log(S))$, the protocol is public coin, has $O(\ell S\log(T))$ rounds, $O(\ell S\log(T)\log(S))$ bits of communication, and perfect completeness.

Proof. The idea is that the verifier first chooses some ϵ bias sets, D, by Lemma 5.7, which is good for M, the adjacency matrix of the computation graph, with high probability. If it isn't good, the prover uses Lemma 5.12. Otherwise the prover uses Lemma 5.10 to reduce to a claim about $\widehat{M}$, which the verifier can calculate itself using Theorem 3.25, and a claim about $\widehat{D}$, which can be verified with Theorem 4.2.

To be more specific, let $L \in \mathbf{NTISP}[T,S]$ be a language recognized by some nondeterministic algorithm A running in simultaneous time T and space S. Assume $T = 2^t$ is a power of 2. On an input x, let M be the adjacency matrix of the computation graph of A on input x. Let a be the start state of A and b be the unique accept state of A. Then A accepts if and only if

$$M^{(T)}(u,v) = 1.$$

Note that $M^{(2)}$ is just 2^{2S} ORs over 2^S variables. So for a field of characteristic 2, D being good for M up to time T is equivalent to D being good for $\log(T)2^{2S}$ different ORs of 2^S variables. By Lemma 5.7, there is an $R = O(S)$ and $L = 2^\ell = O(S)$, so that $D : \{0,1\}^R \times \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ from Theorem 5.6 is not good for all these ORs with probability at most $\frac{1}{6}$. Define $D_r : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ by $D_r(j,w) = D(r,j,w)$. Then equivalently, the probability that D_r is not good for M up to time T, is at most $\frac{1}{6}$.

Let $C = O(S)$ be the amount of space that D runs in so that the soundness of Theorem 4.2 on D_r is $\frac{(4C+2S)\log(T)}{|\mathbb{F}|}$.

Let $\mathbb{F}$ be a field of characteristic 2 with size $|\mathbb{F}| \geq 6\log(T)((\ell+3)(6S+2)+4C)$ and $|\mathbb{F}| < 12\log(T)((\ell+3)(6S+2)+4C)$. Since S and T are efficiently computable, the verifier can calculate $|\mathbb{F}|$ efficiently. The verifier first selects $r \in \{0,1\}^R$ and sends it to the prover. Next the prover sends back a claim that either D_r is good or D_r is bad.

If the prover claims D_r is good, the verifier runs Lemma 5.10 on the claim that $M^{(T)}(u,v) = 1$, and gets back the claim that for some $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ that $\widehat{M}(u',v') = \alpha'$ and $\widehat{D}(j',w') = \beta'$. If

the prover claims D_r is bad, the verifier runs Lemma 5.12 to reduce the claim that D is bad for M up to time T to the claim that for some $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.

The verifier uses Theorem 3.25 to calculate $\widehat{M}(u', v')$ and rejects if it is not equal to α'. Then the verifier runs Theorem 4.2 to verify if $\widehat{D}(j', w') = \beta'$.

The verifier time is just the sum of the time of these subroutines, which is just

$$\begin{aligned}
&R + (S\ell \log(T) + n + S + n + S\log(T))\tilde{O}(\log(|\mathbb{F}|)) \\
=&(n + S\log(S)\log(T))\tilde{O}(\log(S)) \\
=&(n + S\log(T))\tilde{O}(\log(S)^2).
\end{aligned}$$

The verifier space is just the maximum space of any subroutine, which is just

$$O((S + \ell)\log(|\mathbb{F}|)) = O(S\log(S)).$$

For the prover time, the prover must first compute both $M^{(2^k)}$ and $M_D^{(2^k)}$ for every $k \in \log(T)$ to check if D is good. This takes time $2^{O(\ell+S)}$. Then the prover runs either Lemma 5.10, or Lemma 5.12, either of which takes time $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$. Finally the prover needs to run Theorem 4.2, which takes time $2^{O(S)}\log(T)\tilde{O}(\log(|\mathbb{F}|))$. So the total prover time is

$$2^{O(\ell+S)} + 2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|)) + 2^{O(S)}\log(T)\tilde{O}(\log(|\mathbb{F}|)) = 2^{O(S)}.$$

For completeness, if D_r is good, then it follows from completeness of Lemma 5.10 and Theorem 4.2. If D_r is not good, then it follows from completeness of Lemma 5.12 and Theorem 4.2.

For soundness, from soundness of Theorem 5.6, with probability at most $\frac{1}{6}$ is D_r bad. If D_r is good, but the prover claims D_r is bad, by soundness of Lemma 5.10, with probability at most

$$\frac{\log(T)(\ell + 3)(6S + 2)}{|\mathbb{F}|} \leq \frac{1}{6}$$

will $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$. If either of these are false, then by soundness of Theorem 4.2, with probability at most

$$\frac{(4C + 2S)\log(T)}{|\mathbb{F}|} \leq \frac{1}{6}$$

will the verifier accept. So by a union bound, the probability the verifier accepts if the prover claims D_r is bad when D_r is good is at most $\frac{1}{3}$.

If D_r is good and the prover claims D_r is good, then if the verifier gives the incorrect claim about the value of $M^T(a, b)$, then by the soundness of Lemma 5.10, with probability at most

$$\frac{\log(T)(\ell + 2)(6S + 2)}{|\mathbb{F}|} \leq \frac{1}{6}$$

will $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$. If either of these are false, then by soundness of Theorem 4.2, with probability at most

$$\frac{(4C + 2S)\log(T)}{|\mathbb{F}|} \leq \frac{1}{6}$$

will the verifier accept. So by a union bound, the probability the verifier accepts if the prover claims D_r is good when D_r is good, but gives an incorrect value for $M^T(a, b)$, is at most $\frac{1}{3}$.

So the probability the verifier outputs the wrong value for $M^T(a, b)$ is the probability the prover convinces the verifier that D_r is bad or that $M^{(T)}{}_{D_r}(a, b)$ is the wrong value. If D_r is good, then either of these options only occur with probability $\frac{1}{3}$, and there is only a $\frac{1}{6}$ probability that D_r is bad. So our protocol only fails with probability $\frac{1}{2}$. □

6 Interactive Proofs For Alternating Algorithms

Now we give our interactive protocols for alternating algorithms. This still uses the same Razborov-Smolensky degree reduction technique used for nondeterministic algorithms to reduce the degree of large fan in AND and ORs. The main conceptual challenge is rewriting the alternating algorithm in the correct format. So we do this first.

6.1 Alternation Reductions For Bounded Space

To prove our interactive protocol with alternating algorithms, we first must convert our algorithm into a simpler, layered algorithm. This is closely related to the reduction from an alternating algorithm to a low depth circuit by Ruzzo and Tompa [SV84], and a similar reduction was used by Fortnow and Lund [FL93] in their interactive proof for alternating algorithms.

Definition 6.1 (M with d Alternations)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ and integer d, define M with time T and d alternations inductively on d as a function $B^d : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ by*

$d = 1$**:** $B^1(u,v) = M(u,v)$.

d **is even**

$$\begin{aligned} B^d(u,v) =& \forall w \in \{0,1\}^S : M(u,w) \implies B^{d-1}(w,v) \\ =& \neg \bigvee_{w \in \{0,1\}^S} (M(u,w) \wedge \neg B^{d-1}(w,v)) \end{aligned}$$

d **is odd**

$$\begin{aligned} B^d(u,v) =& \exists w \in \{0,1\}^S : M(u,w) \wedge B^{d-1}(w,v) \\ =& \bigvee_{w \in \{0,1\}^S} M(u,w) \wedge B^{d-1}(w,v). \end{aligned}$$

Our interactive proof will focus on this intermediate representation of an alternating circuit as a matrix M with d quantifiers of S variables between them.

Lemma 6.2 (Layered Alternating Programs)**.** *For any $L \in \mathbf{ATISP}_d[T,S]$, there is a nondeterministic algorithm A running in time $T' = O(T)$ and space $S' = O(S)$ such that on any input x, if M is the adjacency matrix of the computation graph of A on input x, then $x \in L$ if and only if the $M^{(T')}$ with d alternations, B^d as defined in Definition 6.1, has $B^d(a,b) = 1$ for some unique starting state a and unique accepting state b.*

Proof. The idea is to modify the alternating algorithm for L, call it A', to keep a timer and synchronize when quantifiers are alternated. Then instead of quantifying over all the state transitions themselves, we only quantify over the states that can be reached between alternations.

If A' ends with a for all quantifier instead of a there exists quantifier, swap the unique accept state with the unique reject state and use the same protocol. So we assume A' ends with an existential quantifier.

So for every alternation, our new algorithm A takes a starting state u, simulates A' for up to time T, and if A' changes quantifiers, we wait until the timer reaches time T before continuing. Then A only run for the $T' = O(T)$ steps between our quantifiers is our new algorithm. The only space required for A is the space for A', plus $\log(T) = O(S)$ for a timer.

Let c be the extra space used by A for a counter at time T (and the same counter for time 0), in addition to anything else needed by A for book keeping. See that $B^1((c,u),(c,v)) = 1$ if and only if A can transition from state u to state v in time T with no change in alternation. By assumption, A' ends in an existential

quantifier, so if u is the unique start state and v is the unique accept state with $d = 1$, then $x \in L$ if and only if for $a = (c, u)$ and $b = (c, v)$, we have $B^1(a, b) = 1$.

More generally, if u is the unique start state for A' and v the unique accept state for A', then $a = (c, u)$ is the unique start for A and $b = (c, v)$ is the unique end state for A. Then one can show by induction that we have $B^d(a, b) = 1$ if and only if

- For even d, for any state w_d reachable from u (in A') within time T without changing quantifiers and the quantifier changes to existential at state w_d, then by induction continuing on from state w_1, algorithm A' will accept w_1 using $d - 1$ alternations.
- For odd d, there exists a state w_1 reachable from u (in A') within time T without changing quantifiers such that w_1 changes to a for all and algorithm A' will accept w_1 using $d - 1$ alternations.

□

6.2 Interactive Proof For Layered Alternations

Now the rest of the proof closely follows the proof for nondeterministic algorithms, defining M with alternations relative to D, showing how an alternation reduction for M with alternations relative to D, and a protocol to show that D is bad.

A subtle difference is that our interactive protocols actually reduce a statement about our alternating algorithm, to a statement about $M^{(T)}$, where M is the adjacency matrix of a nondeterministic algorithm. So we then have to apply our interactive proofs for nondeterministic algorithms. That is, we reduce our statement about alternating algorithms to one about nondeterministic ones, which we already developed the tools for.

Definition 6.3 (M with d Alternations, Relative to D)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, and $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$, we define M with d alternations, relative to D, inductively on d as a function $B_D^d : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ by*

$d = 1$**:** $B_D^1(u, v) = M(u, v)$.

d **is even** $B_D^d(u, v) = \prod_{k \in \{0,1\}^\ell}(1 + \sum_{w \in \{0,1\}^S} D_r(k, w)(M(u, w) + M(u, w)B_D^{d-1}(w, v)))$ mod 2.

d **is odd** $B_D^d(u, v) = 1 + \prod_{k \in \{0,1\}^\ell}(1 + \sum_{w \in \{0,1\}^S} D_r(k, w)(M(u, w)B_D^{d-1}(w, v)))$ mod 2.

Now we show a protocol for reducing the alternations by one.

Lemma 6.4 (IP for M with d Alternations, Relative To D, Single Step)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ and integer $d > 1$, $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ let $B_D^d : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be M with d layered alternations relative to D, as defined in Definition 6.3. Let $\widehat{B_D^d}$ be the multilinear extension of B_D^d.*

Then there is an $O(\ell S)$ round interactive protocol with $O(\ell S \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S \tilde{O}(\log(|\mathbb{F}|))$, space $O((\ell + S)\log(|\mathbb{F}|))$, and a prover P (given the truth table of M, B_D^d and D) that runs in time $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{B_D^d}(u, v) = \alpha$, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta', \gamma' \in \mathbb{F}$ such that $\widehat{B_D^{d-1}}(w', v') = \alpha'$, $\widehat{D}(j', w') = \beta'$, and $\widehat{M}(u', w') = \gamma'$.*

Soundness: *If $\widehat{B_D^d}(u, v) \neq \alpha$, then for any prover P' with probability at most $\frac{(\ell+1)(6S+1)}{|\mathbb{F}|}$ will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta', \gamma' \in \mathbb{F}$ such that $\widehat{B_D^{d-1}}(w', v') = \alpha'$, $\widehat{D}(j', w') = \beta'$, and $\widehat{M}(u', w') = \gamma'$.*

Proof. The proof is very similar to that of Lemma 5.9. We first use the extended product reduction Lemma 5.3, then use the sum check Lemma 3.26 to get a claim about $\widehat{D}$, $\widehat{M}$, and $\widehat{B^{d-1}}_D$. The proof is similar for even or odd d. So we will just prove this for the case that d is even.

More explicitly, the verifier gets the claim that $\widehat{B^d_D}(u,v) = \alpha$. Let $f : \{0,1\}^\ell \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be defined by

$$f(k,u^*,v^*) = 1 + \sum_{w\in\{0,1\}^S} D_r(k,w)(M(u^*,w) + M(u^*,w)B^{d-1}_D(w,v^*)).$$

Recall that for any binary u^*, v^*, we have that

$$\begin{aligned} B^d_D(u^*,v^*) &= \prod_{k\in\{0,1\}^\ell} (1 + \sum_{w\in\{0,1\}^S} D_r(k,w)(M(u^*,w) + M(u^*,w)B^{d-1}_D(w,v^*))) \\ &= \prod_{k\in\{0,1\}^\ell} f(k,u^*,v^*). \end{aligned}$$

Then we apply Lemma 5.3 to reduce to the claim that for some $j' \in \mathbb{F}$, $u, v' \in \mathbb{F}^S$ and α_1 we have

$$\begin{aligned} \alpha^1 =& \widehat{f}(j',u',v') \\ =& 1 + \sum_{w\in\{0,1\}^S} \widehat{D_r}(j',w)(\widehat{M}(u',w) + \widehat{M}(u',w)\widehat{B^{d-1}_D}(w,v')) \\ 1 + \alpha^1 =& \sum_{w\in\{0,1\}^S} \widehat{D_r}(j',w)(\widehat{M}(u',w) + \widehat{M}(u',w)\widehat{B^{d-1}_D}(w,v')). \end{aligned}$$

To see that this formula for $\widehat{f}$ is correct, observe it agrees with f on binary inputs, and is multilinear in j', u' and v'. Now we just run a sum check on this individual degree 3 function in w to get the claim that for some $w' \in \mathbb{F}^S$ and α_2 we have

$$\alpha_2 = \widehat{D_r}(j',w')(\widehat{M}(u',w') + \widehat{M}(u',w')\widehat{B^{d-1}_D}(w',v')).$$

Finally, the prover asks for $\widehat{B^{d-1}_D}(w',v') = \alpha'$, $\widehat{D}(j',w') = \beta'$, and $\widehat{M}(u',w') = \gamma'$ and checks if

$$\alpha_2 = \beta'(\gamma' + \gamma'\alpha').$$

The verifier time and space is just the verifier time and space from Lemma 5.3 and Lemma 3.26, which matches the claim. The same for the prover time. Completeness holds by completeness of Lemma 5.3 and Lemma 3.26.

For soundness, see that Lemma 5.3 gives a soundness of $\frac{l(6S+1)}{|\mathbb{F}|}$, and Lemma 3.26 gives a soundness of $\frac{3S}{|\mathbb{F}|}$. This gives a total soundness of at most

$$\frac{(\ell+1)(6S+1)}{|\mathbb{F}|}$$

. □

Applying this many times gives an interactive protocol reducing a statement about our alternating algorithm to one about a nondeterministic one, which we can solve using the ideas in Theorem 5.13.

Lemma 6.5 (IP for M with d Alternations, Relative To D)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ and integer d, $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ let $B^d_D : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be M with d layered alternations relative to D, as defined in Definition 6.3. Let $\widehat{B^d_D}$ be the multilinear extension of B^d_D.*

Then there is an $O(\ell Sd)$ round interactive protocol with $O(\ell Sd\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell Sd\tilde{O}(\log(|\mathbb{F}|))$, space $O((\ell+S)\log(|\mathbb{F}|))$, and a prover P (given the truth table of M, B^d_D and D) that runs in time $d2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{B^d_D}(u,v) = \alpha$, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.*

Soundness: *If $\widehat{B^d_D}(u,v) \neq \alpha$, then for any prover P' with probability at most $\frac{d(\ell+2)(6S+2)}{|\mathbb{F}|}$ will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.*

Proof. Similar to Lemma 5.10, we will apply Lemma 6.4 for d times, applying multi point reductions Lemma 3.28 on $\widehat{M}$ and $\widehat{D}$ at each step to keep the space down.

Thus the verifier only needs to run Lemma 6.4 and Lemma 3.28 $O(d)$ times, this gives the claimed time, noting the space can be reused. The prover similarly only needs to run Lemma 6.4 and Lemma 3.28 $O(d)$ times.

Completeness holds by completeness of Lemma 6.4 and Lemma 3.28. For soundness, we note that the soundness of Lemma 6.4 is $\frac{(\ell+1)(6S+1)}{|\mathbb{F}|}$, which only needs to be performed d times. And the soundness of Lemma 3.28 on $\widehat{M}$ is $\frac{2S}{|\mathbb{F}|}$ and on $\widehat{D}$ is $\frac{S+\ell}{|\mathbb{F}|}$, which only needs to be performed d times. Thus total soundness is

$$d\frac{(\ell+1)(6S+1)+2S+(S+\ell)}{|\mathbb{F}|} = \frac{d(\ell+2)(6S+2)}{|\mathbb{F}|}.$$

□

This would be enough if D was always good, but now let us formally define what it means for D to be good and show how to handle when it is not.

Definition 6.6 (D is good for M up to d Alternations)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, d and $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$, we say that D is good for M with up to d alternations if for all $k \in [d]$ with $k > 1$ we have $B^k_D = B^k$.*

But when D is bad, we give a protocol showing where it is bad.

Lemma 6.7 (Proving D is Bad for M with Alternations)**.** *For some $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, and integer d, let $\widehat{M}$ be the multilinear extension of M. Let ℓ be an integer and $D : \mathbb{F}^\ell \times \mathbb{F}^S \to \mathbb{F}$ a multilinear function.*

Then there is a round $O(\ell S d)$ interactive protocol with $O(\ell S d \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S d \tilde{O}(\log(|\mathbb{F}|))$ and space $O((\ell+S)\log(|\mathbb{F}|))$, and a prover P (with access tot he truth table of M) that runs in time $d2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ such that

Completeness: *If D is not good for M up to d alternations, then when V interacts with P, V outputs $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.*

Soundness: *If D is good for M up to d alternations, then when V interacts with P, with probability at most $\frac{d(\ell+2)(6S+2)}{|\mathbb{F}|}$ will V output $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.*

Proof. The proof is similar to Lemma 5.12. We take the smallest k such that D is not good for M up to k, show where D is bad at, and then run Lemma 6.5 to prove the value of M with d alternations at that location.

Specifically,

- the prover provides $d' \leq d$, and $u, v, w \in \mathbb{F}^S$ with the claims that

$$B^{d'}_D(u,v) \neq B^{d'}(u,v),$$
$$B^{d'-1}_D = B^{k-1},$$

if d' is even

$$\begin{aligned} B^{d'}(u,v) &= 0, \\ B_D^{d'}(u,v) &= 1, \\ M(u,w)(1-B^{d'-1}(w,v)) &= 1, \end{aligned}$$

and if d' is odd

$$\begin{aligned} B^{d'}(u,v) &= 1, \\ B_D^{d'}(u,v) &= 0, \\ M(u,w)B^{d'-1}(w,v) &= 1. \end{aligned}$$

- We use Lemma 6.4 to reduce the claim about $B_D^{d'}(u,v)$ to a claim that for some $u_1, v_1, w_1 \in \mathbb{F}^S$, $j_1 \in \mathbb{F}^\ell$, and $\alpha_1, \beta_1, \gamma_1 \in \mathbb{F}$ that $\widehat{B_D^{d'-1}}(w_1, v_1') = \alpha_1$, $\widehat{D}(j_1, w_1) = \beta_1$, and $\widehat{M}(u_1, w_1) = \gamma_1$.
- Then we use the multi-point reduction, Lemma 3.28, to reduce the claims about $B_D^{d'-1}$ to the claim that for some $u_2, v_2 \in \mathbb{F}^S$ and $\alpha_2 \in \mathbb{F}$ we have $B_D^{d'-1}(u_2, v_2) = \alpha_2$.
- Then we apply Lemma 6.5 to reduce this to the claim that for some $u_3, v_3, w_3 \in \mathbb{F}^S$, $j_3 \in \mathbb{F}^\ell$, and $\alpha_3, \beta_3 \in \mathbb{F}$ such that $\widehat{M}(u_3, v_3) = \alpha_3$ and $\widehat{D}(j_3, w_3) = \beta_3$.
- Finally we apply Lemma 3.28 again to reduce the claims about $\widehat{M}$ to the claim that for some $u', v' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ we have $\widehat{M}(u', v') = \alpha'$.

  Similarly, we use Lemma 3.28 again to reduce the claims about $\widehat{D}$ to the claim that for some $j' \in \mathbb{F}^\ell$, $w' \in \mathbb{F}^S$ and $\beta; \in \mathbb{F}$ we have $\widehat{D}(j', w') = \beta'$.

The verifier time is just the sum of the times for Lemma 6.4, Lemma 6.5, and Lemma 3.28, which is $\ell S d \tilde{O}(\log(|\mathbb{F}|))$. Similarly, the verifier space is just $O((\ell + S)\log(|\mathbb{F}|))$. The prover only needs to compute each B^i and B_D^i and run the lemmas, which takes time $d2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$.

For completeness, if d' is the smallest such that D is bad for M up to d' alternations, then by definition $B_D^{d'} \neq B^k$ and $B_D^{d'-1} = B^{k-1}$. Thus there is some $u, v \in \{0,1\}^S$ such that $B_D^{d'}(u,v) \neq B^{d'}(u,v)$. Now we do two cases based on whether d' is even.

d' is even: then $B^{d'}(u,v) = \forall w \in \{0,1\}^S : M(u,w) \implies B^{d'-1}(w,v)$. So if $B^{d'}(u,v) = 1$, for any $w \in \{0,1\}^S$, we must have $M(u,w)(1 - B^{d'-1}(w,v)) = 0$. Thus

$$\begin{aligned} B_D^{d'}(u,v) &= \prod_{k\in\{0,1\}^\ell}(1 + \sum_{w\in\{0,1\}^S} D_r(k,w)(M(u,w) + M(u,w)B_D^{d'-1}(w,v))) \mod 2 \\ &= \prod_{k\in\{0,1\}^\ell}(1 + \sum_{w\in\{0,1\}^S} D_r(k,w)0) \mod 2 \\ &= 1. \end{aligned}$$

Thus $B_D^{d'}(u,v) = B^{d'}(u,v)$, a contradiction. Thus $B^{d'}(u,v) = 0$ and $B_D^{d'}(u,v) = 1$. Since $B^{d'}(u,v) = 0$, there must be a w such that $M(u,w)(1 - B^{d'-1}(w,v)) = 1$.

d' is odd: then $B^{d'}(u,v) = \exists w \in \{0,1\}^S : M(u,w) \wedge B^{d'-1}(w,v)$. So if $B^{d'}(u,v) = 0$, for any $w \in \{0,1\}^S$, we must have $M(u,w)B^{d'-1}(w,v) = 0$. Thus

$$\begin{aligned} B_D^{d'}(u,v) &= 1 + \prod_{k\in\{0,1\}^\ell}(1 + \sum_{w\in\{0,1\}^S} D_r(k,w)(M(u,w)B_D^{d'-1}(w,v))) \mod 2 \\ &= 1 + \prod_{k\in\{0,1\}^\ell}(1 + \sum_{w\in\{0,1\}^S} D_r(k,w)0) \mod 2 \\ &= 0. \end{aligned}$$

Thus $B_D^{d'}(u,v) = B^{d'}(u,v)$, a contradiction. Thus $B^{d'}(u,v) = 1$ and $B_D^{d'}(u,v) = 0$. Since $B^{d'}(u,v) = 1$, there must be a w such that $M(u,w)B^{d'-1}(w,v) = 1$.

Thus the prover can provide the requested d', u, v and w. Then the rest of completeness follows from the completeness of Lemma 6.4, Lemma 6.5, and Lemma 3.28.

For soundness, see that if D is good for M up to d alternations, then $B_D^{d'}(u,v) = B^{d'}(u,v)$. So either the claimed value of $B^{d'}(u,v)$ is wrong, or the claimed value of $B_D^{d'}(u,v)$ is wrong. If the claimed value of $B_D^{d'}(u,v)$ is wrong, then by soundness of Lemma 6.4, with probability at most $\frac{(\ell+1)(6S+1)}{|\mathbb{F}|}$ will $\widehat{B_D^{d'-1}}(w_1,v_1) = \alpha_1$, $\widehat{D}(j_1,w_1) = \beta_1$, and $\widehat{M}(u_1,w_1) = \gamma_1$.

If the claimed value of $B^{d'}(u,v)$ is wrong, then the claimed value of $M(u,w)$ or $B^{d'-1}(w,v) = B_D^{d'-1}(w,v)$ is wrong. If the claimed value of $B_D^{d'-1}(w,v)$ is wrong, or $\widehat{B_D^{d'-1}}(w_1,v_1) \neq \alpha_1$, then by Lemma 3.28, with probability at most $\frac{2S}{|\mathbb{F}|}$ will $B_D^{d'-1}(u_2,v_2) = \alpha_2$.

If If the claimed value of $B_D^{d'-1}(w_2,v_2) \neq \alpha_2$, then by soundness of Lemma 6.5, with probability at most $\frac{(d-1)(\ell+2)(6S+2)}{|\mathbb{F}|}$ will $\widehat{M}(u_3,v_3) = \alpha_3$ and $\widehat{D}(j_3,w_3) = \beta_3$.

If the claimed value of $M(u,v)$ is wrong, $\widehat{M}(u_1,w_1) \neq \gamma_1$, $\widehat{D}(j_1,w_1) \neq \beta_1$, or $\widehat{D}(j_3,w_3) \neq \beta_3$, then by soundness of Lemma 3.28, with probability at most $\frac{2S+\ell}{|\mathbb{F}|}$ will $\widehat{M}(u',v') = \alpha'$ and $\widehat{D}(j',w') = \beta'$.

So the probability the verifier accepts that $\widehat{M}(u',v') = \alpha'$ and $\widehat{D}(j',w') = \beta'$ is at most

$$\frac{(\ell+1)(6S+1) + 2S + (d-1)(\ell+2)(6S+2) + 2S + \ell}{|\mathbb{F}|} = \frac{d(\ell+2)(6S+2)}{|\mathbb{F}|}.$$

□

Finally, we can prove our main theorem. Again, we see that the polylogarithmic factor overhead is $\tilde{O}(\log(S)^2)$. As noted in the nondeterministic section, this is worse than the $\tilde{O}(\log(S))$ factor overhead for deterministic algorithms in [Coo22b].

Theorem 1.1 (Interactive Proof For Alternating Space)**.** *For any T, S, and d constructible in time $O(S\log(T))$ and space $O(S)$:*

$$\mathbf{ATISP}_d[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}\left(n + S\log(T) + Sd\right), 2^{O(S)}\right].$$

Further, the verifier runs in space $O(S\log(d+S))$, the protocol is public coin, has $O(S\log(S)(\log(T)+d))$ rounds, $O(S\log(S)(\log(T)+d)\log(d+S))$ bits of communication, and perfect completeness.

Proof. First, because of Lemma 6.2, we can assume that our algorithm comes in the form of a layered, alternating program defined by some nondeterministic algorithm A, which for any x has a computation graph with adjacency matrix M. Specifically, we let T be a power of 2, and let $M^{(T)}$ with d alternations be B_d as defined in Definition 6.1. Specifically, $x \in L$ if and only if for some unique start state u and unique end state v we have $B_d(u,v) = 1$.

Then our protocol (with an honest prover) is quite simple:

1. First we need to sample r so that D_r with high probability is good for M up to time T and good for $M^{(T)}$ up to d alternations. This is equivalent to D being good for $\log(T)+d$ ORs, since being good for M up to time T is just $\log(T)$ ORs, and being good for $M^{(T)}$ up to d alternations is just $\log(T)$ ORs.

   By Lemma 5.7, there is an $R = O(S)$ and $L = 2^\ell = O(S)$, so that $D : \{0,1\}^R \times \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ from Theorem 5.6 is not good for all these ORs with probability at most $\frac{1}{6}$. Thus with probability at most $\frac{1}{6}$ will $D_r : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ defined by $D_r(j,w) = D(r,j,w)$ not be good for M up to time T and good for $M^{(T)}$ up to d alternations.

Let $C = O(S)$ be the amount of space that D runs in so that the soundness of Theorem 4.2 on D_r is $\frac{(4C+2S)\log(T)}{|\mathbb{F}|}$.

Let $\mathbb{F}$ be a field of characteristic 2 with size $|\mathbb{F}| \geq 6(d+\log(T))((\ell+3)(6S+2)+4C)$ and $|\mathbb{F}| < 12(d+\log(T))((\ell+3)(6S+2)+4C)$. Since S and T are efficiently computable, the verifier can calculate $|\mathbb{F}|$ efficiently. The verifier first selects $r \in \{0,1\}^R$ and sends it to the prover. Next the prover sends back a claim that either D_r is good or D_r is bad.

2. If D_r is bad for M up to time T,
   (a) We use Lemma 5.12 to reduce the claim that D_r is bad to the claim that $\widehat{M}(u',v') = \alpha'$ and $\widehat{D_r}(j',w') = \beta'$.
   (b) The verifier computes $\widehat{M}(u',v')$ directly and checks if $\widehat{M}(u',v') = \alpha'$.
   (c) We use use Theorem 4.2 to check if $\widehat{D_r}(j',w') = \beta'$.
3. If D_r is bad for $M^{(T)}$ up to d alternations,
   (a) We use Lemma 6.7 to reduce the claim that D_r is bad to the claim that $\widehat{M^{(T)}}(u',v') = \alpha'$ and $\widehat{D_r}(j',w') = \beta'$.
   (b) Then we run Lemma 5.10 to reduce the claim that $\widehat{M^{(T)}}(u',v') = \alpha'$ to the claim that $\widehat{M}(u'',v'') = \alpha''$ and $\widehat{D_r}(j'',w'') = \beta''$.
   (c) The verifier computes $\widehat{M}(u'',v'')$ directly and checks if $\widehat{M}(u'',v'') = \alpha''$.
   (d) We use Lemma 3.28 to reduce the claim that $\widehat{D_r}(j',w') = \beta'$ and $\widehat{D_r}(j'',w'') = \beta''$ to the claim that $\widehat{D_r}(j''',w''') = \beta'''$.
   (e) Finally, we use Theorem 4.2 to check if $\widehat{D_r}(j''',w''') = \beta'''$.
4. Otherwise, D_r is good and we run Lemma 6.5 to reduce to the claim that $\widehat{M^{(T)}}(u',v') = \alpha'$ and $\widehat{D_r}(j',w') = \beta'$.
5. Then we run Lemma 5.10 to reduce the claim that $\widehat{M^{(T)}}(u',v') = \alpha'$ to the claim that $\widehat{M}(u'',v'') = \alpha''$ and $\widehat{D_r}(j'',w'') = \beta''$.
6. The verifier computes $\widehat{M}(u'',v'')$ directly and checks if $\widehat{M}(u'',v'') = \alpha''$.
7. We use Lemma 3.28 to reduce the claim that $\widehat{D_r}(j',w') = \beta'$ and $\widehat{D_r}(j'',w'') = \beta''$ to the claim that $\widehat{D_r}(j''',w''') = \beta'''$.
8. Finally, we use Theorem 4.2 to check if $\widehat{D_r}(j''',w''') = \beta'''$.

The verifier time is just the sum of these subroutine times, which is just

$$\begin{aligned}&(S\ell d + S\ell\log(T)S + n + S\log(T))\tilde{O}(\log(|\mathbb{F}|))\\ =&(n + S\log(S)(\log(T)+d))\tilde{O}(\log(S))\\ =&(n + S\log(T) + Sd)\tilde{O}(\log(S)^2).\end{aligned}$$

Verifier space is the max of the space used, which is just $O(R + (S+\ell)\log(|\mathbb{F}|)) = O(S\log(|\mathbb{F}|))$. The prover time is just the sum of the times of the sub protocols, which is just $(d+\log(T))2^{O(S+\ell)}\tilde{O}(\log(|\mathbb{F}|)) = 2^{O(S)}$.

The completeness follows from all of the sub protocols. For soundness, if D_r is good, then the probability the prover can convince the verifier that D_r is bad for M up to time T is at most $\frac{\log(T)(\ell+3)(6S+2)}{|\mathbb{F}|} \leq \frac{1}{6}$.

Similarly, the probability the prover can convince the verifier that D_r is bad for $M^{(T)}$ up to time T is at most $\frac{d(\ell+2)(6S+2)}{|\mathbb{F}|} < \frac{1}{6}$.

If the prover does not claim D_r is bad when D_r is not bad, but $x \notin L$ then the probability that the verifier accepts is at most

$$\begin{aligned}
&\frac{d(\ell+2)(S+2)+\log(T)(\ell+2)(6S+2)+\ell+S+(4C+2S)\log(T)}{|\mathbb{F}|} \\
=&(d+\log(T))\frac{(\ell+3)(6S+3)+4C}{|\mathbb{F}|} \\
\leq&\frac{1}{6}.
\end{aligned}$$

So by a union bound, the probability the verifier accepts is at most $\frac{1}{3}$. □

7 Open Problems

While this does mostly close the gap between what the best verifier times between deterministic and nondeterministic algorithms are, many interesting open problems remain, including:

1. Finding a stronger relationship between verifier time (or even alternating time) and bounded space. We know, for $S \geq n$, that
$$\mathbf{TISP}[T,S] \subseteq \mathbf{ITIME}[\tilde{O}(S\log(T))].$$
But it is unknown whether even with the seemingly stronger class of alternating algorithms if
$$\mathbf{TISP}[T,S] \subseteq \mathbf{ATIME}[o(S\log(T))].$$
A major open question is whether the above relationship is true.

2. Finding a stronger relationship between verifier time and alternating time. We know, for $T \geq n$, that
$$\mathbf{ATISP}[T,S] \subseteq \mathbf{ITIME}[\tilde{O}(ST)].$$
Can this factor of S be removed?

3. Giving interactive protocols for $\mathbf{BPTISP}[T,S]$ with simultaneous verifier time $\tilde{O}(n+S\log(T))$, prover time $2^{O(S)}$ *and* perfect completeness. Cook [Coo22b] gave a protocol with that verifier and prover time, but with imperfect completeness.

   Perfect completeness can be achieved in a black box way [Für+89], but these black box reductions do not preserve the prover time.

   One could also reduce to alternating algorithms and use our protocols for alternating algorithms, but known reductions are not efficient enough to get this verifier time.

4. Better doubly efficient proofs. In our special case of alternating algorithms, we can not get provers who run in less than exponential time, without giving sub-exponential time deterministic algorithms for nondeterministic problems.

   But even in the deterministic time and space bounded setting, for $S \geq n$, a major open problem is whether
$$\mathbf{TISP}[T,S] \subseteq \mathbf{ITIME}[\text{poly}(S),\text{poly}(T)].$$
We do know from [GKR15] that
$$\mathbf{TISP}[T,S] \subseteq \mathbf{ITIME}[\text{poly}(S),2^{O(S)}],$$
and from [RRR16] that
$$\mathbf{TISP}[T,S] \subseteq \mathbf{ITIME}[T^{o(1)},\text{poly}(T)],$$
but it is unknown if both the fast verifier time and prover time can be achieved simultaneously.

5. Similar verifier time for algorithms with more general kinds of quantifiers. For instance, threshold quantifiers.

   Currently the most verifier efficient known interactive protocol for threshold circuits is to use shallow circuits to compute threshold and run GKR. In this paper, we showed one can do better for unbounded fan-in AND and OR gates. Can this also be done for unbounded fan-in threshold gates? This would be interesting because threshold gates seem much more powerful than AND, OR, or parity gates.

Acknowledgments

Thanks to Dana Moshkovitz for introducing the authors of this paper, Anna Gál for finding the references relating low depth circuits to alternating algorithms, and Edward Hirsch for references to related interactive proofs for deterministic algorithms.

Joshua Cook is supported by the National Science Foundation under grant number 2200956. Ron Rothblum is funded by the European Union (ERC, FASTPROOF, 101041208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. 1st. USA: Cambridge University Press, 2009. ISBN: 0521424267.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic Checking of Proofs: A New Characterization of NP". In: *J. ACM* 45.1 (Jan. 1998), 70–122. ISSN: 0004-5411. DOI: 10.1145/273865.273901. URL: https://doi.org/10.1145/273865.273901.

[Alo+90] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. "Simple construction of almost k-wise independent random variables". In: *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. 1990, 544–553 vol.2. DOI: 10.1109/FSCS.1990.89575.

[Aro+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof Verification and the Hardness of Approximation Problems". In: *J. ACM* 45.3 (May 1998), 501–555. ISSN: 0004-5411. DOI: 10.1145/278298.278306. URL: https://doi.org/10.1145/278298.278306.

[BFL90] L. Babai, L. Fortnow, and C. Lund. "Nondeterministic exponential time has two-prover interactive protocols". In: *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. 1990, 16–25 vol.1. DOI: 10.1109/FSCS.1990.89520.

[BM00] David Mix Barrington and Alexis Maciel. "Basic Course on Computational Complexity. Lecture 14: IP = PSPACE". In: IAS/PCMI Summer Session 2000. Clay Mathematics Undergraduate Program, 2000. URL: http://people.clarkson.edu/~alexis/PCMI/Notes/lectureB14.ps.gz (visited on 08/21/2023).

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Theory of Cryptography Conference*. 2016.

[Bab+91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking Computations in Polylogarithmic Time". In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. STOC '91. New Orleans, Louisiana, USA: Association for Computing Machinery, 1991, 21–32. ISBN: 0897913973. DOI: 10.1145/103418.103428. URL: https://doi.org/10.1145/103418.103428.

[Bor77] Allan Borodin. "On Relating Time and Space to Size and Depth". In: *SIAM Journal on Computing* 6.4 (1977), pp. 733–744. DOI: 10.1137/0206054. eprint: https://doi.org/10.1137/0206054. URL: https://doi.org/10.1137/0206054.

[Bus87] S. R. Buss. "The Boolean Formula Value Problem is in ALOGTIME". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. New York, New York, USA: Association for Computing Machinery, 1987, 123–131. ISBN: 0897912217. DOI: 10.1145/28395.28409. URL: https://doi.org/10.1145/28395.28409.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. "Alternation". In: *J. ACM* 28.1 (1981), 114–133. ISSN: 0004-5411. DOI: 10.1145/322234.322243. URL: https://doi.org/10.1145/322234.322243.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. "Practical Verified Computation with Streaming Interactive Proofs". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, 90–112. ISBN: 9781450311151. DOI: 10.1145/2090236.2090245. URL: https://doi-org.ezproxy.lib.utexas.edu/10.1145/2090236.2090245.

[Coo22a] Joshua Cook. "More Verifier Efficient Interactive Protocols for Bounded Space". In: Electronic Colloquium on Computational Complexity (ECCC), 2022. URL: https://eccc.weizmann.ac.il/report/2022/093/.

[Coo22b] Joshua Cook. "More Verifier Efficient Interactive Protocols for Bounded Space". In: *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*. Ed. by Anuj Dawar and Venkatesan Guruswami. Vol. 250. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 14:1–14:18. ISBN: 978-3-95977-261-7. DOI: 10.4230/LIPIcs.FSTTCS.2022.14. URL: https://drops.dagstuhl.de/opus/volltexte/2022/17406.

[Coo79] Stephen A. Cook. "Deterministic CFL's Are Accepted Simultaneously in Polynomial Time and Log Squared Space". In: *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*. STOC '79. Atlanta, Georgia, USA: Association for Computing Machinery, 1979, 338–345. ISBN: 9781450374385. DOI: 10.1145/800135.804426. URL: https://doi.org/10.1145/800135.804426.

[Din+99] Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. "PCP Characterizations of NP: Towards a Polynomially-Small Error-Probability". In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*. STOC '99. Atlanta, Georgia, USA: Association for Computing Machinery, 1999, 29–40. ISBN: 1581130678. DOI: 10.1145/301250.301265. URL: https://doi.org/10.1145/301250.301265.

[FL92] Uriel Feige and László Lovász. "Two-Prover One-Round Proof Systems: Their Power and Their Problems (Extended Abstract)". In: *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '92. Victoria, British Columbia, Canada: Association for Computing Machinery, 1992, 733–744. ISBN: 0897915119. DOI: 10.1145/129712.129783. URL: https://doi.org/10.1145/129712.129783.

[FL93] Lance Fortnow and Carsten Lund. "Interactive Proof Systems and Alternating Time-Space Complexity". In: *Theor. Comput. Sci.* 113.1 (1993), 55–73. ISSN: 0304-3975. DOI: 10.1016/0304-3975(93)90210-K. URL: https://doi.org/10.1016/0304-3975(93)90210-K.

[FSS81] Merrick Furst, James B. Saxe, and Michael Sipser. "Parity, circuits, and the polynomial-time hierarchy". In: *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*. 1981, pp. 260–270. DOI: 10.1109/SFCS.1981.35.

[Fei+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. "Approximating Clique is Almost NP-Complete (Preliminary Version)". In: *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*. IEEE Computer Society, 1991, pp. 2–12. DOI: 10.1109/SFCS.1991.185341. URL: https://doi.org/10.1109/SFCS.1991.185341.

[Für+89] Martin Fürer, Oded Goldreich, Y. Mansour, Michael Sipser, and Stathis Zachos. "On Completeness and Soundness in Interactive Proof Systems". In: *Adv. Comput. Res.* 5 (1989), pp. 429–442.

[GG79] Ofer Gabber and Zvi Galil. "Explicit constructions of linear size superconcentrators". In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. 1979, pp. 364–370. DOI: 10.1109/SFCS.1979.16.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *J. ACM* 62.4 (Sept. 2015). ISSN: 0004-5411. DOI: 10.1145/2699436. URL: https://doi.org/10.1145/2699436.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM J. Comput.* 18.1 (1989), pp. 186–208. DOI: 10.1137/0218012. URL: https://doi.org/10.1137/0218012.

[GR20] Oded Goldreich and Guy N. Rothblum. "Constant-Round Interactive Proof Systems for AC0[2] and NC1". In: *Computational Complexity and Property Testing: On the Interplay Between Randomness and Computation*. Cham: Springer International Publishing, 2020, pp. 326–351. ISBN: 978-3-030-43662-9. DOI: 10.1007/978-3-030-43662-9_18. URL: https://doi.org/10.1007/978-3-030-43662-9_18.

[Gol18] Oded Goldreich. *On Doubly-Efficient Interactive Proof Systems*. 2018. URL: https://www.wisdom.weizmann.ac.il/~oded/de-ip.html.

[HMS13] Edward Hirsch, Dieter van Melkebeek, and Alexander Smal. "Succinct Interactive Proofs for Quantified Boolean Formulas, Comment 2". In: Electronic Colloquium on Computational Complexity (ECCC), 2013. URL: https://eccc.weizmann.ac.il/report/2012/077/comment/2/download/.

[KR08] Yael Tauman Kalai and Ran Raz. "Interactive PCP". In: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II*. ICALP '08. Reykjavik, Iceland: Springer-Verlag, 2008, 536–547. ISBN: 9783540705826. DOI: 10.1007/978-3-540-70583-3_44. URL: https://doi.org/10.1007/978-3-540-70583-3_44.

[Kah95] Nabil Kahale. "Eigenvalues and expansion of regular graphs". In: *Journal of the ACM* 42.5 (1995), pp. 1091–1106. ISSN: 0004-5411.

[Lun+90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. "Algebraic methods for interactive proof systems". In: *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. 1990, 2–10 vol.1. DOI: 10.1109/FSCS.1990.89518.

[MBIS90] David A. Mix-Barrington, Neil Immerman, and Howard Straubing. "On Uniformity within NC1". In: *J. Comput. Syst. Sci.* 41.3 (1990), 274–306. ISSN: 0022-0000. DOI: 10.1016/0022-0000(90)90022-D. URL: https://doi.org/10.1016/0022-0000(90)90022-D.

[MW18] Cody Murray and Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2018. Los Angeles, CA, USA: Association for Computing Machinery, 2018, 890–901. ISBN: 9781450355599. DOI: 10.1145/3188745.3188910. URL: https://doi.org/10.1145/3188745.3188910.

[Mar73] Grigorii Aleksandrovich Margulis. "Explicit construction of concentrators". Trans. by Problemy Peredachi Informatsii. In: *Problemy Peredachi Informatsii* 9 (1973), pp. 325–332. URL: https://cir.nii.ac.jp/crid/1572824500389149056.

[Mei13] Or Meir. "IP = PSPACE Using Error-Correcting Codes". In: *SIAM Journal on Computing* 42.1 (2013), pp. 380–403. DOI: 10.1137/110829660. eprint: https://doi.org/10.1137/110829660. URL: https://doi.org/10.1137/110829660.

[NN93] Joseph Naor and Moni Naor. "Small-Bias Probability Spaces: Efficient Constructions and Applications". In: *SIAM Journal on Computing* 22.4 (1993), pp. 838–856. DOI: 10.1137/0222053. eprint: https://doi.org/10.1137/0222053. URL: https://doi.org/10.1137/0222053.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. "Constant-Round Interactive Proofs for Delegating Computation". In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Cambridge, MA, USA: Association for Computing Machinery, 2016, 49–62. ISBN: 9781450341325. DOI: 10.1145/2897518.2897652. URL: https://doi.org/10.1145/2897518.2897652.

[Raz05] Ran Raz. "Quantum Information and the PCP Theorem". In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '05. USA: IEEE Computer Society, 2005, 459–468. ISBN: 0769524680. DOI: 10.1109/SFCS.2005.62. URL: https://doi.org/10.1109/SFCS.2005.62.

[Raz87] Alexander A. Razborov. "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition". In: *Mathematical notes of the Academy of Sciences of the USSR* 41 (1987), pp. 333–338.

[Ruz81] Walter L. Ruzzo. "On uniform circuit complexity". In: *Journal of Computer and System Sciences* 22.3 (1981), pp. 365–383. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(81)90038-6. URL: https://www.sciencedirect.com/science/article/pii/0022000081900386.

[SV84] Larry Stockmeyer and Uzi Vishkin. "Simulation of Parallel Random Access Machines by Circuits". In: *SIAM Journal on Computing* 13.2 (1984), pp. 409–422. DOI: 10.1137/0213027. eprint: https://doi.org/10.1137/0213027. URL: https://doi.org/10.1137/0213027.

[San07] Rahul Santhanam. "Circuit Lower Bounds for Merlin-Arthur Classes". In: *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. STOC '07. San Diego, California, USA: Association for Computing Machinery, 2007, 275–283. ISBN: 9781595936318. DOI: 10.1145/1250790.1250832. URL: https://doi.org/10.1145/1250790.1250832.

[Sha92] Adi Shamir. "IP = PSPACE". In: *J. ACM* 39.4 (Oct. 1992), 869–877. ISSN: 0004-5411. DOI: 10.1145/146585.146609. URL: https://doi.org/10.1145/146585.146609.

[She92] A. Shen. "IP = SPACE: Simplified Proof". In: *J. ACM* 39.4 (1992), 878–880. ISSN: 0004-5411. DOI: 10.1145/146585.146613. URL: https://doi.org/10.1145/146585.146613.

[Smo87] R. Smolensky. "Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. New York, New York, USA: Association for Computing Machinery, 1987, 77–82. ISBN: 0897912217. DOI: 10.1145/28395.28404. URL: https://doi.org/10.1145/28395.28404.

[TS17] Amnon Ta-Shma. "Explicit, Almost Optimal, Epsilon-Balanced Codes". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. Montreal, Canada: Association for Computing Machinery, 2017, 238–251. ISBN: 9781450345286. DOI: 10.1145/3055399.3055408. URL: https://doi-org.ezproxy.lib.utexas.edu/10.1145/3055399.3055408.

[TV02] L. Trevisan and S. Vadhan. "Pseudorandomness and average-case complexity via uniform reductions". In: *Proceedings 17th IEEE Annual Conference on Computational Complexity*. 2002, pp. 129–138. DOI: 10.1109/CCC.2002.1004348.

[Tha13] Justin Thaler. "Time-Optimal Interactive Proofs for Circuit Evaluation". In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 71–89.

[Tha20] Justin Thaler. *The Unreasonable Power of the Sum-Check Protocol*. 2020. URL: https://zkproof.org/2020/03/16/sum-checkprotocol/ (visited on 07/01/2023).

[Tha22] Justin Thaler. "Proofs, Arguments, and Zero-Knowledge". In: *Found. Trends Priv. Secur.* 4.2-4 (2022), pp. 117–660. DOI: 10.1561/3300000030. URL: https://doi.org/10.1561/3300000030.

[Vad12] Salil P. Vadhan. "Pseudorandomness". In: *Foundations and Trends® in Theoretical Computer Science* 7.1–3 (2012), pp. 1–336. ISSN: 1551-305X. DOI: 10.1561/0400000010. URL: http://dx.doi.org/10.1561/0400000010.

[Wra76] Celia Wrathall. "Complete sets and the polynomial-time hierarchy". In: *Theoretical Computer Science* 3.1 (1976), pp. 23–33. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(76)90062-1. URL: https://www.sciencedirect.com/science/article/pii/0304397576900621.

[Xie+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. Lecture Notes in Computer Science. Springer, 2019, pp. 733–764. DOI: 10.1007/978-3-030-26954-8_24. URL: https://doi.org/10.1007/978-3-030-26954-8_24.

[Zha+21] Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time". In: *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. Ed. by Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi. ACM, 2021, pp. 159–177. DOI: 10.1145/3460120.3484767. URL: https://doi.org/10.1145/3460120.3484767.

A Arithmetization of Nondeterministic Algorithm

For completeness, we include lemma 36 from [Coo22a], generalized to also be a multilinear extension of the input.

Theorem 3.25 (Arithmetization of State Transition)**.** *Suppose A is a space S nondeterministic algorithm with transition matrix $M_n : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ as described in Definition 3.22.*

Then we can compute the multilinear extension of M_n, denoted $\widehat{M_n} : \mathbb{F}^n \times \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$, in time $(n+S)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(S)+\log(n))\log(|\mathbb{F}|))$.

Proof. The idea is to write the transition function as a sum over the transition rules of the multilinear extension of that rule being followed. Then we write each transition rule as a product of several components which use disjoint variables. Then we calculate the multilinear extension of each of these components efficiently. Finally, we note that the sum of the products is multilinear, and agrees with M_n on binary inputs, so is the multilinear extension of M_n.

From the definition of the state of a Turing Machine Definition 3.21, we can represent a state $s^0 \in \{0,1\}^S$ as $s^0 = (q^0, h^0, w^0, m^0)$ where q^0 is the instruction state, h^0 is the index of the input tape head, w^0 is the index of the working tape head, and m^0 is the contents of the working tape. And from the definition of a

Turing Machine Transition matrix, Definition 3.22, for any $x \in \{0,1\}^n$ and $s^1 = (q^1, h^1, w^1, m^1)$ we have

$$\begin{aligned} M_n(x, s^0, s^1) = 1 \equiv \exists\, ((q_0, \sigma_h, \sigma_s), (q_1, d_h, d_w, \sigma_f) = \lambda \in \Lambda : \\ q_0 = q^0 \wedge \\ q_1 = q^1 \wedge \\ \sigma^h = x_{h^0} \wedge \\ h^0 + d_h = h^1 \wedge \\ m^0_{w^0} = \sigma_s \wedge \\ m^1_{w^0} = \sigma_f \wedge \\ w^0 + d_w = w^1 \wedge \\ \forall j \in [S'] \setminus \{w^0\} : m^0_j = m^1_j. \end{aligned}$$

Now we will give a claimed formula for $\widehat{M_n}$ and then show that it is correct. We claim that

$$\begin{aligned} \widehat{M_n}(x, s^0, s^1) = \sum_{((q_0,\sigma_h,\sigma_s),(q_1,d_h,d_w,\sigma_f))=\lambda\in\Lambda} & \operatorname{equ}(q_0, q^0)\operatorname{equ}(q_1, q^1) \\ & \operatorname{input}_\lambda(x, h^0, h^1) \\ & \operatorname{work}_\lambda(w^0, m^0, w^1, m^1) \end{aligned}$$

where $\operatorname{input}_\lambda$ checks that the input and input head is consistent with λ while $\operatorname{work}_\lambda$ checks that the work head and the working tape is consistent with λ. Specifically,

$$\begin{aligned} \operatorname{input}_\lambda(x, h^0, h^1) = \sum_{i\in[n'], i+d_h\in[n']} & \operatorname{equ}(i, h^0)\operatorname{equ}(i + d_h, h^1)\operatorname{equ}(x_i, \sigma_h) \\ \operatorname{work}_\lambda(w^0, m^0, w^1, h^1) = \sum_{i\in[S'], i+d_w\in[S']} & \operatorname{equ}(i, w^0)\operatorname{equ}(i + d_w, w^1) \\ & \operatorname{equ}(m^0_i, \sigma_s)\operatorname{equ}(m^1_i, \sigma_f) \\ & \operatorname{equ}(m^0_{[i-1]}, m^1_{[i-1]}) \\ & \operatorname{equ}(m^0_{[S]\setminus[i]}, m^1_{[S]\setminus[i]}). \end{aligned}$$

Now see that $\operatorname{work}_\lambda$ is multilinear, since for each term in the sum, each variable only appears in one equ. Similarly, $\operatorname{input}_\lambda$ is multilinear, and $\widehat{M_n}$ is multilinear.

Now we argue that on binary inputs $\widehat{M_n}$ is equal to M_n. To see this, see that $M_n(s^0, s^1) = 1$ if and only if there exists some $\lambda \in \Lambda$ such that all the equalities from Definition 3.22 hold. But these equalities are exactly the equalities enforced by the corresponding term in $\widehat{M_n}$. Further, each $\lambda \in \Lambda$ is mutually exclusive, so only at most one term in the sum is 1 on binary inputs.

Thus $\widehat{M_n}$ is the multilinear extension of M_n. Now it remains to show how to calculate $\widehat{M_n}$ efficiently. Note that since there are only constantly many $\lambda \in \Lambda$, it suffices to show how to compute $\operatorname{input}_\lambda$ and $\operatorname{work}_\lambda$ efficiently.

The hardest part of $\operatorname{input}_\lambda$ is $\operatorname{equ}(i, h^0)$. To calculate it directly each time it is used would take $n \log(n)$ field operations, which is more than we claimed.

To calculate $\operatorname{equ}(i, h^0)$ efficiently, for each $j \in [\log(n)]$ calculate $\operatorname{equ}(i_{[j]}, h^0_{[j]})$, where the last bit in i is the least significant bit. Note that for any $j < \log(n)$ we can efficiently calculate $\operatorname{equ}(i_{[j+1]}, h^0_{[j+1]})$ from $\operatorname{equ}(i_{[j]}, h^0_{[j]})$. Then in expectation, as we increment i, we only change $i_{[j]}$ for constantly many j. Thus we can iterate through each $\operatorname{equ}(i, h^0)$ using in $O(n)$ field operations storing only $O(\log(n))$ field elements. Similarly for $\operatorname{equ}(i + d_h, h^1)$.

Thus $\operatorname{input}_\lambda$ can be calculated in time $\tilde{O}(\log(|\mathbb{F}|))n$ and space $O(\log(|\mathbb{F}|)\log(n))$.

For work$_\lambda$, we handle $\mathrm{equ}(i, w^0)\mathrm{equ}(i+d_w, w^1)$ similar to input$_\lambda$. The hard terms are $\mathrm{equ}(m^0_{[i-1]}, m^1_{[i-1]})$ and $\mathrm{equ}(m^0_{[S]\setminus[i]}, m^1_{[S]\setminus[i]})$. If we recalculated these directly each time they were used, this would take $O(S^2)$ field operations. Instead, we iteratively calculate each $\mathrm{equ}(m^0_{[i-1]}, m^1_{[i-1]})$ from $\mathrm{equ}(m^0_{[i-2]}, m^1_{[i-2]})$, which only requires one field operation. So each $\mathrm{equ}(m^0_{[i-1]}, m^1_{[i-1]})$ can be calculated with $O(S)$ field operations and only storing $O(1)$ field elements.

Now for $\mathrm{equ}(m^0_{[S]\setminus[i]}, m^1_{[S]\setminus[i]})$, we have some technical details. We calculate this for each i starting from 0, up to S. So we cannot get $\mathrm{equ}(m^0_{[S]\setminus[i]}, m^1_{[S]\setminus[i]})$ from $\mathrm{equ}(m^0_{[S]\setminus[i-1]}, m^1_{[S]\setminus[i-1]})$ from a multiplication. Instead, we have to use a division:

$$\mathrm{equ}(m^0_{[S]\setminus[i]}, m^1_{[S]\setminus[i]}) = \frac{\mathrm{equ}(m^0_{[S]\setminus[i-1]}, m^1_{[S]\setminus[i-1]})}{\mathrm{equ}(m^0_i, m^1_i)}$$

If no $\mathrm{equ}(m^0_i, m^1_i) = 0$, then this also takes $O(S)$ field operations and only requires storing $O(1)$ field elements.

But now we must worry about potential division by 0. But we would only come across this possibility if for some j we have $\mathrm{equ}(m^0_j, m^1_j) = 0$. If this is true, then for any $i > j$ we have $\mathrm{equ}(m^0_{[i-1]}, m^1_{[i-1]}) = 0$, so that term in the sum is 0, and for any $i < j$ we have $\mathrm{equ}(m^0_{[S]\setminus[i]}, m^1_{[S]\setminus[i]}) = 0$. Thus the only term in the sum potentially non zero is when $i = j$, so we can just calculate that term directly using $O(S)$ field operations and only storing $O(1)$ field elements.

So work$_\lambda$ can be calculated in time $\tilde{O}(\log(|\mathbb{F}|))S$ and space $O(\log(|\mathbb{F}|)\log(S))$. Thus altogether $\widehat{M_n}$ can be calculated in time $(n+S)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(n)+\log(S))\log(|\mathbb{F}|))$. □

B Standard Algebraic Interactive Proof Tools

In this section, we include proofs of many of the theorems in Section 3.6. First we establish that polynomial interpolation can be computed quickly. This is done essentially by defining the interpolating polynomials, showing how to calculate them quickly, and showing how to interpolate with them. First, we define our interpolating polynomials.

Definition B.1 (Interpolating Polynomials)**.** *For any m, and $i \in [m]$, define $\ell^m_i : \mathbb{F} \to \mathbb{F}$ to be the unique degree 1 polynomial such that $\ell^m_i(i) = 1$ and for all other $j \in [m] \setminus \{i\}$ we have $\ell^m_i(j) = 0$.*

Now we state that we can calculate them quickly. A proof can be found in [Coo22b].

Lemma B.2 (Calculating Interpolating Polynomials Fast)**.** *For any integer m , and for any $x \in \mathbb{F}$, one can calculate the sequence $(\ell^m_i(x))_{i\in[m]}$ in time $\tilde{O}(\log(|\mathbb{F}|))m$ and space $O(\log(|\mathbb{F}|))$.*

For any integer m and sequence of vectors $(w_i \in \mathbb{F}^S)_{i\in[m]}$, for any $x \in [m]$, one can calculate

$$g(x) = \sum_{i\in[m]} w_i \ell^m_i(x)$$

in time $mS\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|))$.

Now we state that the above formula is enough to do fast polynomial interpolation.

Corollary B.3 (Interpolating Polynomials Fast)**.** *For any integer d degree d polynomial $g : \mathbb{F} \to \mathbb{F}^S$, if one is given for each $i \in [d+1]$ the value $g(i)$, then for any $x \in \mathbb{F}$ one can calculate*

$$g(x) = \sum_{i\in[d+1]} g(i)\ell^{d+1}_i(x)$$

in time $dS\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|))$.

Alternatively, if one is given for each $i \in [0,d]$ the value $g(i)$, then for any $x \in \mathbb{F}$ one can calculate

$$g(x) = \sum_{i \in [0,d]} g(i) \ell_i^{d+1}(x+1)$$

in time $dS\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S \log(|\mathbb{F}|))$.

Now we begin the proofs from Section 3.6. Starting with unlinearization, which is a straightforward application of sum check.

Lemma 3.27 (Unlinearization)**.** *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ is a polynomial with individual degree d, and $\widehat{f}$ is the multilinear function consistent with f on binary inputs. Then there is an $S+1$ round interactive protocol with $O(dS \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $Sd\tilde{O}(\log(|\mathbb{F}|))$ and space $O((d+S)\log(|\mathbb{F}|))$, and a prover P that runs in time $d2^S\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(d2^S)$ oracle queries to f which takes as input a $w \in \mathbb{F}^S$ and $\alpha \in \mathbb{F}$ such that*

Completeness: *If $\widehat{f}(w) = \alpha$, then when V interacts with P, V outputs a $w' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ such that $f(w') = \alpha'$.*

Soundness: *If $\widehat{f}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{(d+1)S}{|\mathbb{F}|}$ will V output a w' and α' such that $f(w') = \alpha'$.*

Proof. Observe that

$$\widehat{f}(w) = \sum_{y \in \{0,1\}^S} \mathrm{equ}(w,y)\tilde{f}(y)$$

since $\tilde{f}$ agrees with f on binary inputs and both sides are multilinear.

Than applying a sum check reduces the claim that

$$\widehat{f}(w) = \alpha$$

to the claim that for some $\beta \in \mathbb{F}$ and $w' \in \mathbb{F}^S$ that

$$\beta = \mathrm{equ}(w,w')f(w').$$

We ask the prover to also provide α', which should be equal to $f(w')$. The verifier then calculates $\mathrm{equ}(w,w')$ and rejects if $\beta \neq \mathrm{equ}(w,w')\alpha'$

Since $g(y) = \mathrm{equ}(w,y)\tilde{f}(y)$ only has individual degree $d+1$, the sum check has soundness $\frac{(d+1)S}{|\mathbb{F}|}$. See that if $\beta \neq \mathrm{equ}(w,w')f(w')$, and the prover provides $\alpha' = f(w')$, the verifier rejects. So this protocol also has soundness $\frac{(d+1)S}{|\mathbb{F}|}$. There is only one more round than sum check, and only a $O(\log(|\mathbb{F}|))$ bit message. Similarly, the verifier time, space, and prover time are the same as sum check, plus a small time and space to calculate equ. □

Now for multi-point reduction.

Lemma 3.28 (Multi-Point Reduction)**.** *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ has total degree d and m is some integer. Then there is a one round interactive protocol with $O(dm \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(S+d)m\tilde{O}(\log(|\mathbb{F}|))$ and space $O((md+S)\log(|\mathbb{F}|))$, and a prover P that runs in time $m^2dS\tilde{O}(\log(|\mathbb{F}|))$ with $O(dm)$ oracle queries to f. The protocol takes as input $(w_i \in \mathbb{F}^S)_{i \in [m]}$ and $(\alpha_i \in \mathbb{F})_{i \in [m]}$ and behaves such that*

Completeness: *If for each $i \in [m]$ we have $f(w_i) = \alpha_i$, then when V interacts with P, V outputs a $w' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ such that $f(w') = \alpha'$.*

Soundness: *If for any $i \in [m]$ we have $f(w_i) \neq \alpha_i$, then for any prover P' with probability at most $\frac{d(m-1)}{|\mathbb{F}|}$ will V output a w' and α' such that $f(w') = \alpha'$.*

Proof. The idea is to take a degree $m-1$ polynomial, $\phi : \mathbb{F} \to \mathbb{F}^S$ that goes through each w_i, and ask the prover for $f \circ \phi$, which is a degree $d(m-1)$ polynomial. This gives a claim from the prover of what f should be at each w_i. If any of these $f(w_i)$ are wrong, then the polynomial provided by the prover must not be $f \circ \phi$. Then by Schwartz-Zippel, they agree on most random locations. Since the verifier can compute ϕ, this gives us our new w'.

More specifically, define $\phi : \mathbb{F} \to \mathbb{F}^S$ as the unique degree $m-1$ polynomial so that for $i \in [m]$ (where we associate integers with elements of $\mathbb{F}$ in any natural way) we have $\phi(i) = w_i$. This degree d polynomial is

$$\phi(x) = \sum_{i \in [m]} w_i \ell_i^m(x)$$

where ℓ_i^m are the interpolation polynomials in Definition B.1. Then by Lemma B.2 the verifier can calculate $\phi(x)$ in time $mS\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|) + \log(m))$.

Then the prover gives a degree $d' = d(m-1)+1$ polynomial, $g : \mathbb{F} \to \mathbb{F}$, which the verifier expects to be $f \circ \phi$. The polynomial g is specified by its value at every $i \in [d']$, and then g can be calculated by polynomial interpolation in time $d'\tilde{O}(\log(|\mathbb{F}|))$ and space $O(\log(|\mathbb{F}|) + \log(d'))$.

Notably, by its representation, if $g = f \circ \phi$, then for each $i \in [m]$ we should have

$$g(i) = f(\phi(i)) = f(w_i) = \alpha_i.$$

If for any $i \in [m]$, $g(i) \neq \alpha_i$, the verifier rejects. Otherwise, the verifier chooses a random $x \in \mathbb{F}$, sets $w' = \phi(x)$, and sets $\alpha' = g(x)$.

By the representation of g, the verifier gets each $g(i)$ for $i \in [m]$ directly, so can check them all in time $O(m\log(|\mathbb{F}|))$ and space $O(\log(|\mathbb{F}|)+\log(S)+\log(m))$. Just to store g, the verifier uses space $O(md\log(|\mathbb{F}|))$. By Lemma B.2 the verifier can calculate $\phi(x)$ in time $mS\tilde{O}(\log(|\mathbb{F}|))$ and space $O(S\log(|\mathbb{F}|) + \log(m))$. Similarly, by Lemma B.2, the verifier can calculate $g(x)$ in time $md\tilde{O}(\log(|\mathbb{F}|))$ and space $O(\log(|\mathbb{F}|) + \log(md))$. So the overall verifier time is $(S+d)m\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+md)\log(|\mathbb{F}|))$.

The prover only needs to evaluate $f \circ \phi$ at d' locations and send those to the verifier. This requires the prover to calculate ϕ at $O(dm)$ locations. Each calculation of ϕ can be performed in time $mS\tilde{O}(\log(|\mathbb{F}|))$. So calculating $\phi(i)$ for each $i \in [d']$ only takes time $dm^2S\tilde{O}(\log(|\mathbb{F}|))$.

For completeness, suppose that if for each $i \in [m]$ we have $f(w_i) = \alpha_i$. Then for each $i \in [m]$ the prover will return $g(i) = f(\phi(i)) = f(w_i) = \alpha_i$. So the verifier doesn't reject. Further since g and $f \circ \phi$ are both degree $d(m-1)$ and agree on $d(m-1)+1$ locations, by thw Schwartz-Zippel lemma they are equal. So in particular, $\alpha' = g(x) = f(\phi(x)) = f(w')$. So we have completeness.

For soundness, suppose for any $i \in [m]$ we have $f(w_i) \neq \alpha_i$. Then if the verifier does not reject, we must have $g(i) \neq f(w_i) = f(\phi(i))$. Thus $g \neq f \circ g$. Then since both are degree $d(m-1)$ polynomials, the agree on at most $d(m-1)$ points. The probability that $g(x) = f(\phi(x))$ is at most $\frac{d(m-1)}{|\mathbb{F}|}$. If $g(x) \neq f(\phi(x))$ then $\alpha' \neq f(w')$. So with probability at most $\frac{d(m-1)}{|\mathbb{F}|}$ will $\alpha' = f(w')$. □

And the special application of multi-point reduction to product reductions.

Lemma 3.29 (Product Reduction)**.** *Suppose $f : \mathbb{F}^S \to \mathbb{F}$ has total degree d. Then there is a one round interactive protocol with $O(d\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(S+d)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+d)\log(|\mathbb{F}|))$, and a prover P that runs in time $Sd\tilde{O}(\log(|\mathbb{F}|))$ and makes $O(d)$ oracle queries to the f. The protocol takes as input $u, v \in \mathbb{F}^S$ and $\alpha \in \mathbb{F}$ and acts such that*

Completeness: *If $f(u)f(v) = \alpha$, then when V interacts with P, V outputs a $w \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ such that $f(w) = \alpha'$.*

Soundness: *If $f(u)f(v) \neq \alpha$, then for any prover P' with probability at most $\frac{d}{|\mathbb{F}|}$ will V output a w and α' such that $f(w) = \alpha'$.*

Proof. This follows directly from the standard multi-point reduction Lemma 3.28. The prover first provides a claimed value for $\alpha_1 = f(u)$ and $\alpha_2 = f(v)$. If $\alpha_1\alpha_2 \neq \alpha$, the verifier rejects. Then we run a multi-point reduction, which asks the verifier to provide f composed with the line that goes through u and v. This verifier and prover runs in the given time and space by Lemma 3.28.

For completeness, if $f(u)f(v) = \alpha$, then an honest prover gives $\alpha_1\alpha_2 = \alpha$. And by completeness of Lemma 3.28, the verifier outputs α' and w such that $f(w) = \alpha'$.

For soundness, if $f(u)f(v) \neq \alpha$, then if $\alpha_1 = f(u)$ and $\alpha_2 = f(v)$, the verifier will reject since $\alpha_1\alpha_2 \neq \alpha$. Otherwise, either $\alpha_1 \neq f(u)$ or $\alpha_2 \neq f(v)$. In either case, by soundness of Lemma 3.28, with probability at most $\frac{d}{|\mathbb{F}|}$ will V output a w and α' such that $f(w) = \alpha'$. □

C Derandomization Proofs

Here we give a proof of the existence of epsilon biased sets, and how to use random walks to sample efficiently.

Lemma 5.4 (ϵ-Biased Set)**.** *For any S, there is an $m = O(S)$ and a function $D' : \{0,1\}^m \times \{0,1\}^S \to \{0,1\}$ such that for any $X \subseteq \{0,1\}^S \setminus \emptyset$*

$$\Pr_{r\in\{0,1\}^m}\left[\sum_{x\in X} D'(r,x) = 1 \pmod 2\right] \geq \frac{1}{4}$$

such that D' runs in poly(S) *time and $O(S)$ space.*

Proof. Note that the existence of D' is exactly equivalent to the existence of a $1/8$ biased set generator for 2^S bits with seed length $O(S)$. As noted by Naor and Naor [NN93], this is equivalent to the construction of a linear code with distance $1/4$ and constant rate. Let the encoder of that linear code be $E : \{0,1\}^{2^S} \times \{0,1\}^{2^m} \to \{0,1\}$. The function D' itself just outputs individual bits of the generating matrices for E. More specifically, for $i \in \{0,1\}^S$ and $j \in \{0,1\}^m$ (which we interpret as integer indexes), we have

$$D'(j,i) = E(e_i)_j$$

where e_i is the vector which is all 0, except a 1 in the ith entry. So our problem reduces to constructing an efficiently computable E.

Perhaps the most straightforward code is the concatenation code of a Reed-Muller code with a Hadamard code. Since both codes are linear, their composition is linear. So our code E will consist of two linear codes, $E = E_2 \circ E_1$, and our j will be interpreted as two indices, $j = (j_1, j_2)$, so that

$$E(e_i)_j = E_2(E_1(i)_{j_i})_{j_2}$$

where E_1 is the Reed-Muller code and E_2 is the Hadamard code.

Specifically let E_1 be the code of degree 2^S polynomials over $\mathbb{F}_{2^{2S}}$ so that $j_1 \in \{0,1\}^{2S}$ which we interpret as $j_1 \in \mathbb{F}_{2^{2S}}$. Specifically,

$$E_1(x)_j = \sum_{i\in[2^S]} x_i j^i.$$

So $E_1(e_i)_j$ is just j^i. This can be calculated by repeated squaring in time poly(S) and space $O(S)$. As a degree 2^S polynomial in a field of size 2^{2S}, by the Schwartz-Zippel lemma, E_1 has relative distance $1 - \frac{2^S}{2^{2S}} > \frac{1}{2}$.

Since E_2 is the Hadamard code, we have that $E_2 : \{0,1\}^{2S} \to \{0,1\}^{2^{2S}}$ such that $E_2(u)_v = u \cdot v$. Thus $j_2 \in \{0,1\}^{2^S}$ and

$$E(e_i)_j = (j_1)^i \cdot j_2$$

where j_1 is interpreted as an element of $\mathbb{F}_{2^{2S}}$, but $(j_1)^i$ is interpreted as an element of $\{0,1\}^{2S}$. We can compute E_2 in space $O(S)$ and time $O(S)$, and E_2 has relative distance of $\frac{1}{2}$. Thus E has a relative distance of δ with $\delta > \frac{1}{4}$ and $D'(j,i) = E_2(E_1(e_i)_{j_1})_{j_2}$ can be computed in time poly(S) and space $O(S)$.

Thus the total distance of E is δ with $\delta > \frac{1}{4}$, and $m = 4S$. Let Δ be the function computing relative hamming distance, that is $\Delta(x, y) = \Pr_i[x_i \neq y_i]$. By the well known correspondence between ϵ biased sets and linear codes [NN93], for any $X \subseteq \{0,1\}^S \setminus \emptyset$ (also interpreted as an element of $\{0,1\}^{2^S}$), we have

$$\begin{aligned}
\Pr_{r\in\{0,1\}^m}\left[\sum_{x\in X} D'(r,x) = 1 \pmod 2\right] &= \Pr_{r\in\{0,1\}^m}\left[\sum_{x\in X} E(e_x)_r = 1 \pmod 2\right] \\
&= \Pr_{r\in\{0,1\}^m}\left[E\Big(\sum_{x\in X} e_x\Big)_r = 1 \pmod 2\right] \\
&= \Pr_{r\in\{0,1\}^m}[E(X)_r = 1 \pmod 2] \\
&= \Delta(E(X), 0) \\
&= \Delta(E(X), E(\emptyset)) \\
&\geq \delta \\
&\geq \frac{1}{4}.
\end{aligned}$$

□

Lemma 5.5 (Efficient Hitting Sampler)**.** *For any m, and any $\epsilon > 0$, there is an $L = 2^\ell = O(\log(1/\epsilon))$, an $R = O(m + L)$ and a function $W : \{0,1\}^R \times \{0,1\}^\ell \to \{0,1\}^m$ such that, if for any $D^* : \{0,1\}^m \to \{0,1\}$ we have that $\Pr_r[D^*(r) = 1] \geq \frac{1}{4}$, then*

$$\Pr_{r\in\{0,1\}^R}[\forall a \in \{0,1\}^\ell : D^*(W(r,a)) = 0] \leq \epsilon.$$

Further, W is computable in time $\mathrm{poly}(R)$ *and space $O(R)$.*

Proof. Our function W is just a random walk on an efficient expander Lemma 3.19, then using the hitting property of random walks on expanders Lemma 3.20.

First if m is not even, add a bit to make it even, m'. If m is already even, set $m' = m$. We can view D^* as a function taking m' bits by ignoring the last bit if $m' > m$. Then see that $|\{0,1\}^{m'}| = n^2$, so there is an expander, G, with size $|\{0,1\}^{m'}|$, constant degree d, and constant spectral expansion $\lambda < 1$. Further, if V are the vertices of G, then G has an edge function, $E : V \times [d] \to V$, that can be computed in space $O(m)$ and time $O(m)$.

Let L be the smallest power of 2 at least L so that $L > \frac{\log(1/\epsilon)}{\log(4)-\log(3+\lambda)}$. See that $L = O(\log(1/\epsilon))$ since λ is constant. Let k be the smallest power of 2 at least d. Then we can describe a length L walk on G using $R = m' + Lk = O(m + L)$ bits.

Then let W be the function which takes an R bit string describing a random walk on G, $w_1, \ldots, w_L$, and an ℓ bit string a time step of that walk, $i \in [L]$, and outputs the vertex in the walk at that time: w_i.

By Lemma 3.19, any w_i can be computed in space $O(m' + \ell) = O(R)$ and time $O(Lm') = O(R^2)$. Let B be the set of vertices in $v \in V$ so that $D^*(v) = 0$. Then $|B| \leq \left|\frac{3}{4}\right|$. So by Lemma 3.20, the probability that all steps in a length L walk on G land in B is

$$\begin{aligned}
\Pr_{r\in\{0,1\}^R}[\forall a \in \{0,1\}^\ell : W(r,a) \in B] &\leq \left(\frac{3+\lambda}{4}\right)^L \\
\Pr_{r\in\{0,1\}^R}[\forall a \in \{0,1\}^\ell : D^*(W(r,a)) = 0] &\leq .
\end{aligned}$$

Now we can derive the stated theorem.

$$\begin{aligned}
\frac{\log(1/\epsilon)}{\log(4) - \log(3+\lambda)} &< L \\
\frac{\log(\epsilon)}{\log(3+\lambda) - \log(4)} &<
\end{aligned}$$

Now using the fact that $\log(3+\lambda) - \log(4) = \log\left(\frac{3+\lambda}{4}\right) < 0$ we have

$$\begin{aligned}
\log(\epsilon) &> \log\left(\frac{3+\lambda}{4}\right) L \\
\epsilon &> \left(\frac{3+\lambda}{4}\right)^{L} \\
&> \Pr_{r\in\{0,1\}^R}[\forall a \in \{0,1\}^\ell : D^*(W(r,a)) = 0].
\end{aligned}$$

□

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il